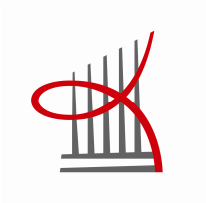


WLAN verification Wizard

Jukka Issakainen

Opinnäytetyö
Joulukuu 2011
Tietotekniikan koulutusohjelma
Tietoliikenteen suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Tietoliikenteen suuntautumisvaihtoehto

ISSAKAINEN, JUKKA: Wlan verification Wizard

Opinnäytetyö 61s., liitteet 36s.
Joulukuu 2011

Langattomien lähiverkkojen yhteensopivuustestauksessa joudutaan käyttämään useita tukiasemia, joilla testattavan tuotteen toiminnallisuus todennetaan. Kullakin tukiasemalla on useita säädettäviä asetuksia ja mitään yhtenäistä standardia käyttöliittymälle ei ole. Tästä voi aiheutua hidastuksia ja virhetilanteita varsinaiseen testaukseen, jos tukiaseman asetukset eivät ole täysin oikein.

Luomalla yksinkertainen valintoihin perustuva käyttöliittymä saadaan tukiasemaohjelmistojen kompleksisuus poistettua testitilanteesta. Ohjelmisto kytkee automaattisesti virrat tukiasemaan, hoitaa yhteydenoton tukiasemaan, lataa varmistustiedostosta halutun kokoonpanon ja määrittää tarvittaessa RADIUS-palvelimen. Tällöin voidaan keskittyä päätehtävään eli testaamiseen.

Ohjelmistolla saavutettiin selkeä parannus tukiasemien määritysvirheistä johtuviin virhetilanteisiin, jotka hidastavat varsinaista testausta ja siten luovat aikataulupaineita. Testauksen luotettavuutta ja toistettavuutta saatiin myös selvästi parannettua.

Tässä lopputyössä keskitytään Wlan verification Wizard – ohjelmiston toiminnallisuuden ja rakenteen kuvaamiseen. Varsinaiseen testaukseen ei paneuduta salassapitosopimusten vuoksi.

Asiasanat: Yhteensopivuustestaus, langaton lähiverkko, tukiasema, asetukset

ABSTRACT

Tampereen ammattikorkeakoulu

Tampere University of Applied Sciences

Degree programme in Computer Science

Option of Datacommunications

ISSAKAINEN, JUKKA: Wlan verification Wizard

Bachelor's thesis 61 pages, appendices 36 pages

December 2011

Interoperability testing for wireless local area networks requires several base stations, also known as access points, which are used to verify the correct operation of a device under test. Each access point has several settings and there is no global standard for user interface. If user has to manually setup each feature, it can cause delays and error situations to actual testing.

By creating a simple user interface based on radio-buttons can the complexity of access point settings be minimized. User interface takes care of powering up the desired access point, communicates with it and restores the desired setup and, if needed, sets the definitions for RADIUS-server. Focus can be set to testing instead of setting up the testing environment.

By using this user interface a clear reduction of errors caused by faulty setups of access points was achieved. These erroneous situations slow down the actual testing and therefore create pressure for schedules. Also reliability and repeatability of testing got better.

This Bachelor's thesis focuses on describing functionality and structure of Wlan verification Wizard. Actual interoperability testing of wireless local area networks is out of scope due to non-disclosure agreements.

Key words: Interoperability testing, wireless lan, access point, settings

SISÄLLYS

1. JOHDANTO

Kun kehitetään tietoliikennetuotteita, on tärkeää ottaa huomioon myös muut markkinoilla olevat tuotteet. Tuotekehityssyklit ovat voimakkaasti lyhentyneet markkinapaineiden vuoksi ja osittain tästä johtuen saattaa markkinoilta löytyä tuotteita, joiden valmistuksessa ei ole seurattu tarkasti alan määritelmiä eli spesifikaatioita tai on jätetty kokonaan huomioimatta perussertifioinnit. Osa näistä sertifioinneista ei ole pakollisia ja se voi johtaa kiusaukseen tuoda markkinoille tuote, jota ei ole testattu sertifiointimääritysten mukaan.

Eräs tällainen sertifiointi on langattomien lähiverkkojen (WLAN) sertifiointi. Wi-Fi Alliance (WFA) (Wi-Fi Alliance Organization 2011) on voittoa tuottamaton järjestö, joka koordinoi langattomien lähiverkkojen kehitystä ja sertifiointia. Se perustettiin vuonna 1999 ja perustajajäseninä olivat 3Com, Aironet, Intersil, Lucent Technologies, Nokia ja Symbol Technologies. Nykyään siihen kuuluu yli 400 jäsenyritystä.

WFA kehittää ja ylläpitää testisuunnitelmia jäsenyritysten tuotteiden sertifiointiin. Yleiset testisuunnitelmat ovat tehtyjä lähinnä pc-keskeisten laitteiden sertifiointia varten. Tästä johtuen jäsenyrityksillä on mahdollisuus käyttää täysin omaa testisuunnitelmaansa, mikäli WFA:n tekninen jaosto hyväksyy sen.

Käytäntö on kuitenkin osoittanut, että pelkkä laitesertifiointi ei takaa riittävää yhteensopivuutta markkinoilla olevien tuotteiden kanssa. Syitä on useita, joista tärkeimpinä:

- sertifiointimääritykset eivät ole aivan ehdottomia, niissä on tulkinnanvaraisuuksia
- kaikki yritykset eivät sertifioi tuotteitaan

Tällöin ainoaksi mahdollisuudeksi jää hoitaa yhteensopivuustestaus käyttäen mahdollisimman montaa eri tuotetta, jonka kanssa yrityksen tuote tulee toimimaan.

Langattomien lähiverkkojen tukiasemia on maailmassa hyvin paljon ja niiden saatavuus myös vaihtelee maantieteellisesti. Myös paikalliset lait voivat vaikuttaa jonkin tuotteen saatavuuteen, esimerkkinä lisensioimattomien radioverkkojen eli niin sanotun Indust-

rial, Scientific, Medical (ISM) – alueen radioverkkojen määräykset. Myös paikallisilla teleoperaattoreilla saattaa olla yksinoikeus tietynmerkkiseen ja – tyyppiseen laitteeseen. Pienen helpotuksen aiheeseen antaa langattomien radioverkkojen piirisarjojen valmistajien suhteellisen pieni määrä. Toki tukiaseman toimintaan vaikuttaa moni muukin seikka, kuten antennisuunnittelu ja tukiasemaohjelmisto, mutta piirisarjan perustoiminnallisuus on kuitenkin kaikissa sama. Tämä seikka huomioonottaen on mahdollista saada kohtuullinen kattavuus tukiasematestauksessa järkevillä kuluilla. Kaikkia tukiasemia ei yksinkertaisesti ole mahdollista hankkia yhteen tietoliikennelaboratorioon.

WFA:n sertifiointitestauksissa käytetään vain muutamaa tukiasemaa ja painotus on suorituskykymittauksissa. Sertifiointitesteissä käydään läpi kaikki tunnistus- eli autentikointitavat, niin yksityis- kuin myös yrityskäyttöön suunnatut metodit.

1.1. Salaustyypit

WFA määrittelee langattomien lähiverkkojen yksityiskäyttöä varten seuraavat tunnistusmenetelmät:

- avoin verkko, eli salausta ei ole
- Wired Equivivalent Privacy (WEP) 64-bittinen, eli 10 heksadesimaalista merkkiä pitkä salasana (poistunut nykyisistä sertifiointivaatimuksista)
- WEP 128-bittinen, eli 26 heksadesimaalista merkkiä pitkä salasana (poistunut nykyisistä sertifiointivaatimuksista)
- Wi-Fi Protected Access (WPA) Personal, joka käyttää TKIP (Temporary Key Integrity Protocol)-salausta (poistunut nykyisistä sertifiointivaatimuksista)
- Wi-Fi Protected Access II (WPA2) Personal, joka käyttää CCMP/AES (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol /Advanced Encryption Standard) –salausta (yleensä käytetään vain AES-nimeä)

Molemmat WEP-salaukset ovat nykyään poistettu sertifiointivaatimuksista, koska ne ovat varsin turvattomia. Samoin WPA on poistettu. Se sisälsi lähes kokonaan IEEE802.11i – standardin ja se oli väliaikainen ratkaisu korvaamaan tietoturvaltaan varsin heikon WEP:n. Tästä huolimatta kaikkia edellämainittuja salausmenetelmiä löy-

tyy lähes jokaisesta langattoman lähiverkon tukiasemasta. Nykyinen WPA2 on varsin turvallinen ja se sisältää täyden tuen IEEE802.11i – standardille.

Yrityskäytössä pelkkä salasanaan pohjautuva tunnistus ei ole riittävä ja silloin käytetäänkin Extensible Authentication Scheme (EAP):a, joka lisää tietoturvaa pohjautumalla esimerkiksi sertifikaattipohjaiseen tunnistukseen. Tällöin tarvitaan myös tunnistuspalvelin. Yksikään tunnistuspalvelin ei toimi kaikkien mahdollisten sertifikaattipohjaisten menetelmien kanssa. Tämän johdosta markkinoilla on useita tuotteita, joista esimerkkeinä Microsoft Server 2003 (Internet Authentication Server IAS)/2008/2008 R2, Cisco ACS ja FreeRADIUS.

Yleisimmät sertifikaattiin perustuvat tunnistusmenetelmät ovat:

- EAP – Transport Layer Security (TLS), Microsoftin kehittämä, IETF RFC 2716
- EAP – Protected Extensible Authentication Protocol (PEAP) v0 ja v1, Microsoftin, Ciscon ja RSA Securityn kehittämä
- EAP – Tunneled Transport Layer Security (TTLS), Funk Softwaren ja Certicomin kehittämä
- EAP – Lightweight Extensible Authentication Protocol (LEAP), Ciscon kehittämä
- EAP – Method for GSM Subscriber Identification Module (SIM), Nokia/Haverinen et al, RFC 4186
- EAP – Method for UMTS Authentication and Key Agreement (AKA), RFC 4187

Tehtävänäni oli hoitaa Nokian matkapuhelinten langattomien lähiverkkojen yhteensopivuustestauksia. Varsin pian kävi selville tarvittavan työn määrä, sillä jokaiselle tukiasemalle oli jokaista testitilannetta varten määriteltävä omat asetuksensa. Kaikista tukiasemista ei suinkaan löytynyt tukea esim sertifikaattipohjaisen tunnistuksen vaatimalle tunnistuspalvelinyhteydelle. Tunnistuspalvelimista käytetään yleisesti nimeä RADIUS, Remote Authentication Dial In User Service. Joskus näkee käytettävän myös lyhennettä AAA (Authentication, Authorization and Accounting).

Lisäksi muiden työkiireiden johdosta jouduin välillä käyttämään ulkopuolista apuvoimaa suorittamaan testauksia. Heillä ei välttämättä aina ollut riittävästi kokemusta mo-

nentyyppisten tukiasemien määrityksistä ja/tai kiire testauksessa. Tämä aiheutti monia turhia vianmetsästyshetkiä ja turhaan kulutettua aikaa.

Pienen helpotuksen asiaan toi mahdollisuus tallentaa sopivat määritykset tiedostoksi ja ladata ne tarvittaessa tukiasemaan. Silti välillä tuntui olevan ongelmia, kun ei välttämättä ymmärretty ladata täysin oikeaa määritystiedostoa, koska ei ymmärretty ajatella kokonaiskuvaa testauksen kannalta. Nämä huomiot saivat aikaan ajatuksen kehittää käyttöliittymä, jonka ainoa tarkoitus olisi varmistaa oikean määritystiedoston lataaminen kuhunkin testihetkellä käytössä olevaan tukiasemaan.

2. WLAN VERIFICATION WIZARD

Aloin miettimään kuinka saisin kohtuullisen nopeasti luotua graafisen käyttöliittymän ja mielellään Windows-ympäristössä toimivana ajettavana tiedostona. Ja mielellään ilman huimia kuluja. Käyttöjärjestelmävalinta oli käytännön sanelema tosiasia, koska monet mittalaitteet, joita käytin kyseisessä testauksessa toimivat Windowsin päällä.

2.1. AutoIt

Muutamien kokeilujen jälkeen päädyin käyttämään täysin ilmaista kehitysympäristöä nimeltä AutoIt (AutoIt: Automation and scripting language 1999 - 2011). Se on BASIC:in tyylinen skriptauskieli Windows-ympäristöön ja se käyttää simuloituja näppäimen painalluksia, hiiren liikkeitä ja suoria Windows-ympäristön ikkunoiden ja kontrollien käyttöjä. Sillä on helppo luoda käyttöliittymä kohtuullisen pienellä vaivalla sekä kääntää luodusta skriptistä Windowsissa toimiva ajonaikainen tiedosto.

Muita hyväksi havaittuja ominaisuuksia ovat laaja Windows – tuki (2000 / XP / 2003 / Vista / 2008 / Windows 7 / 2008 R2), laajahko ja avulias käyttäjäkunta foorumilla, oma editori (viritetty versio SciTE – editorista) ja ennen kaikkea ilmaisuus.

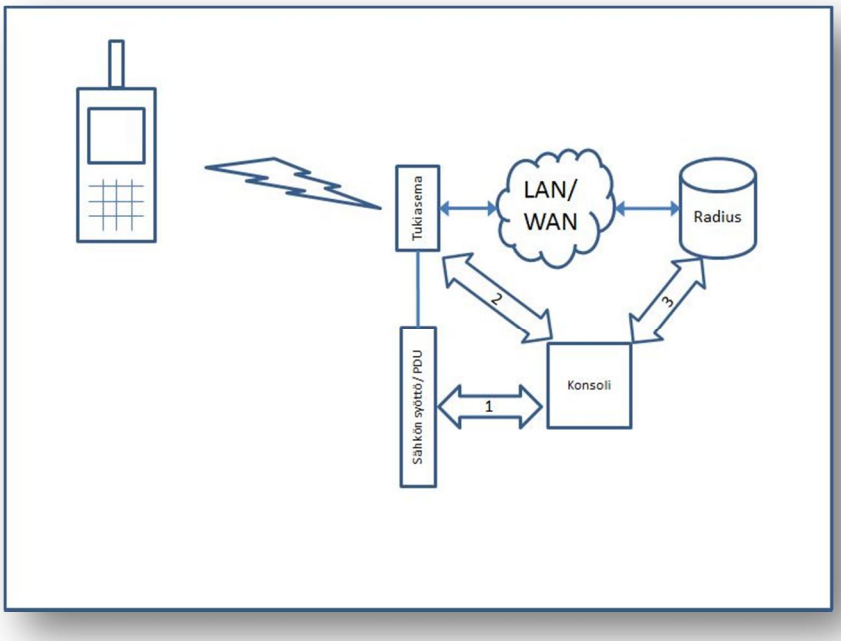
Pääsin siis lopputyössäni opiskelemaan uuden ohjelmointikielen (AutoIt), miettimään hieman käytettävyyttä ja yleisestikin palauttamaan muistiin kuinka ohjelmointia tehdään aikataulupaineiden alla monien muiden projektien ohella. Ohjelman käyttökielenä on englanti, koska alan termistö on pääasiassa englantia ja lähimmät työkaverini eivät puhu suomea. Lisäsin lähdekoodiin kommentteja ja muutenkin yritin ylläpitää kommentoivaa toteutustapaa, jotta mahdollisen vikatilanteen edessä ei tarvitsisi kovin pitkään tutkia koodia, jotta saisi selvää mistä on kyse.

Mitään varsinaista suunnitteludokumentaatiota en katsonut tarpeelliseksi luoda, koska kyseessä oli omaan työkäyttöön tuleva työkaluohjelmisto. Ohjelman ulkoasu muokkautui varsin nopeasti nykyisen kaltaiseksi sisältäen vain välttämättömät elementit.

Normaalisti AutoIt toimii tarkkailemalla kontrolleja (napit, valintaikkunat jne) silmukassa. Tällainen toiminto osoittautui hyvin nopeasti toimimattomaksi, kun ikkunassa olevien kontrollien määrä nousi yli kymmenen. Tällöin ikkunan vaste muuttuu selkeästi

hitaammaksi ja mikäli kontrollien määrä kasvaa, vaste hidastuu vielä entisestään. Onneksi käytettävissä on myös tapahtumanaikainen toiminne eli ”On Event”. Tällöin kontrollin manipulointi (napin painallus, valintarastin asetus jne) palauttaa kyseisen kontrollin yksilöllisen koodin. Se lähetetään viestinä halutulle funktiolle, joka suorittaa määritetyn tehtävän ja palauttaa halutun arvon. Tämän ansiosta pääohjelman ei tarvitse tarkkailla kaikkia kontrolleja jatkuvasti, jolloin käyttöliittymän vasteaika paranee ja ohjelmiston aiheuttama prosessorikuorma pysyy hyvin alhaisena.

2.2. Ohjelman perusidea



Kuva 1: Toiminnallisuuskuvaus

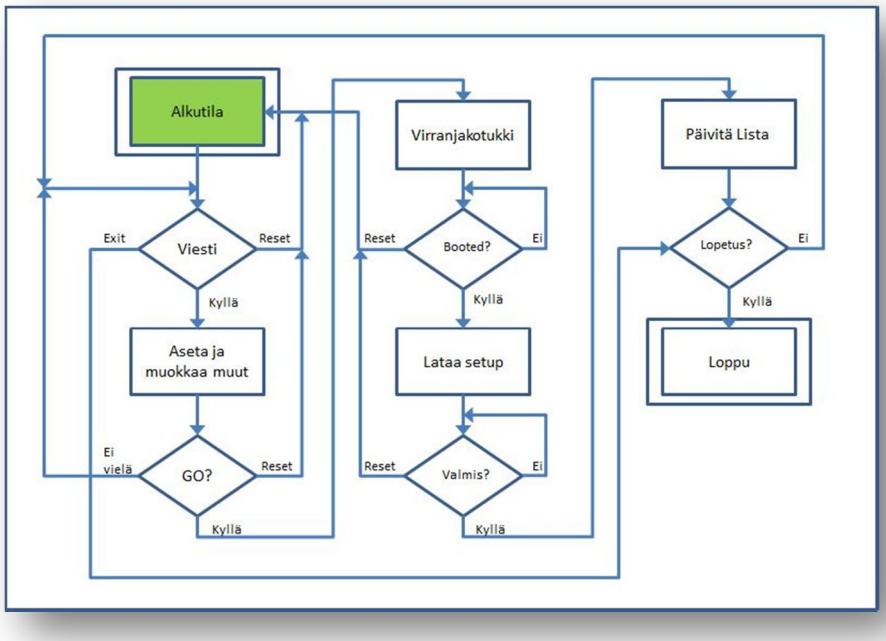
Yllä olevasta kuvasta (Kuva **1**) selviää ohjelman perustoiminnallisuus karkealla tasolla. Wlan verification Wizard – ohjelma toimii erillisellä konsolikoneella. Tarvittaessa sitä voidaan käyttää myös suoraan RADIUS-palvelimen konsolilta, mutta fyysiset rajoitukset saattavat aiheuttaa esteitä. Esimerkkinä mainittakoon käyttämieni RADIUS-palvelinten siirto testitilastani tietoliikennekonehuoneeseen. Nuoli 1 kuvaa sähkön syötön ohjausta, nuoli 2 tukiasemien ohjausta ja nuoli 3 RADIUS-palvelimen ohjausta. Nämä kukin käydään tarkemmin läpi edempänä.

Koska tavoitteena oli luoda työkalu helpottamaan varsin monimutkaista asetusten tekoa, oli selvää että kunkin tukiaseman yksittäisiä asetuksia ei kannata alkaa muokkaamaan. Käyttöönottovaiheessa tallennetaan kultakin tukiasemalta halutulla salausmetodilla oikeiksi todetut asetukset varmuuskopiotiedostoon. Näitä tiedostoja hyväksikäyttäen voidaan olla varmoja, että haluttu salausmetodi pystytään asettamaan tukiasemaan toistettavasti ja ilman virheitä.

Aiemmin mainitsin, että AutoIt simuloi käyttäjää. Wlan verification Wizard -ohjelma komentaa valittua tukiasemaa hakemaan tilanteeseen sopivan asetustiedoston, kuten normaali käyttäjä olisi sen tehnyt. Yhteys tukiasemaan tapahtui useinmiten selaimen avulla; joissain tapauksissa oli mahdollista myös käyttää Telnetiä tai valmistajan erillistä asetusohjelmaa. Vaikkakin AutoIt mahdollistaa suoraan myös näytön objektien manipuloinnin käyttämällä suoria objektiosoitteita havaitsin helpommaksi käyttää tabulointinäppäimen simulointeja, esimerkiksi ”mene viisi tabulointia eteenpäin” ja Enter-näppäimen painalluksia. Tällöin oli mahdollista sopivia näppäinkomentoja käyttäen päästä tekstisyöttökentälle, lähettää kyseiselle tukiasemalle esimerkiksi polku ja tiedostonimi haluttuun asetustiedostoon, hypätä eteenpäin OK-napille ja lähettää napin painallus ohjelmallisesti.

Mielenkiintoisen haasteen muodostivat myös tukiasemien hallintaliittymät. Mitään yhtenäistä standardia ei ole olemassa, vaan jokainen tukiasema on oma yksilönsä, liittymä vaihtelee web-liittymästä omaan päivitysohjelmaan ja tilanne muuttuu tukiaseman ohjelmistopäivitysten mukaan. Tämä aiheuttaa hieman lisätyötä, sillä jokaisen ohjelmistopäivityksen tai selaimen version muuttumisen jälkeen on tarkistettava, että tukiaseman sivuston komponentit ovat niillä kohdilla, mitä ohjelmaan on määritelty.

Seuraavassa kuvassa on yksinkertaistettu ohjelman toimintakaavio. Olen jättänyt selkeyden vuoksi pois monia pienempiä toimintoja, esimerkiksi avautuvan ikkunan tarkistus (onko kyseessä haluttu ikkuna, esim tukiaseman jokin ilmoitusikkuna vai jokin muu). Perustoiminnot ovat kuitenkin selkeästi nähtävissä.



Kuva 2: Yksinkertaistettu toimintakaavio

Ohjelmassani objektin manipulointi (napin painallus, radio-napin valinta jne) luo viestin. Tämä viesti tutkitaan ja sen perusteella suoritetaan haluttu ohjelmoitu funktio. Kullakin objektilla on oma osoitteensa ja toiminteensa objektityypistä riippuen. Tämän jälkeen ohjelma palaa odottamaan seuraavaa viestiä. Kuvassa 2 on Alkutilan jälkeinen valintalaatikko ”Viesti” kuvaamassa tätä tilannetta. Käyttäjä valitsee haluamansa radiotyypin, salauksen, tukiaseman, EAP-tyypin ja RADIUS – palvelimen. Kaikki nämä valinnat ovat dynaamisia eli tehty valinta vaikuttaa muihin valintoihin siten, että estetään mahdottomat tilanteet. Esimerkiksi jos ei ole salausta käytössä, ei myöskään voida valita RADIUS – palvelinta.

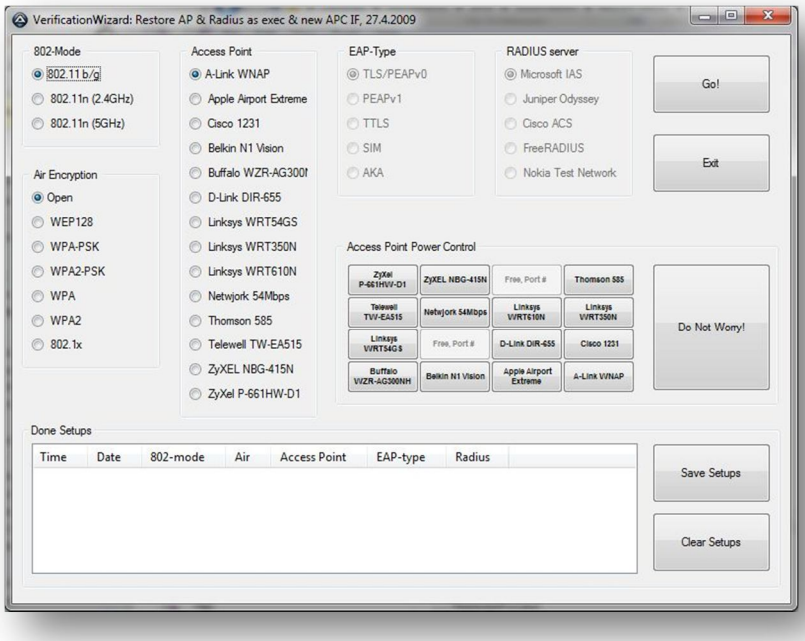
Go-napin painallus aloittaa varsinaisen toiminnan. Ensiksi kytketään sähköt päälle valittuun tukiasemaan. Tämä tapahtuu ottamalla Telnet-yhteys IP-ohjattuun virranjakotukkiin (kuvassa 2 ”Virranjakotukki”). Käyttämässäni versiossa (American Power Conversion AP7950) oli 16 230V ulostuloa, jotka olivat jokainen erikseen ohjattavissa joko selaimella tai Telnetillä. Telnet – yhteys luodaan käyttämällä Plink-ohjelmaa. Plink eli PuTTY Link on Simon Tathamin (Using the command line connection tool Plink 2005) kehittämä ilmainen ja varsin monipuolinen komentoriviohjelma tietoliikennettä varten ja sitä käytetäänkin paljon toimintojen automatisoinnissa.

Koska tukiaseman käynnistyminen vie jonkin verran aikaa, sitä seurataan lähettämällä ping-kyselyitä tukiaseman ip-osoitteeseen (kuvassa 2 APC:n alla oleva ”Booted?” – silmukka). Kun tukiasema vastaa, odotetaan vielä hiukan lisää, sillä tietoliikenneportin toiminnallisuus käynnistyy hieman nopeammin kuin varsinaisen tukiaseman oma toiminnallisuus. Mikäli ping-kysely palauttaa virhekoodin, voidaan sen perusteella ilmoittaa käyttäjälle mahdollisesta kaapeliviasta tai väärästä ip-osoitteesta.

Kun tukiasema on käynnistynyt varmasti, otetaan siihen yhteys joko selaimella tai Telnetillä riippuen halutusta yhteysmuodosta. Tukiaseman hallintasivulla siirrytään varmuuskopion palautukseen ja haetaan haluttu tiedosto (kuvassa 2 ”Lataa setup”). Tämän jälkeen tukiasema käynnistyy uudelleen ja tällöin on käytettävissä valittu salausmetodi.

Mikäli valittu salausmetodi tarvitsee sertifikaattipohjaista tunnistautumista ja sen johdosta yhteyden RADIUS – palvelimeen, käydään tukiaseman käynnistymistä odotellessa tekemässä tarvittavat muutokset RADIUS-välityspalvelimelle netsh-ohjelman (Configure TCP/IP from the Command Prompt 2009) avulla. Tämä on kuvattu tarkemmin kappaleessa RADIUS-PALVELIMEN MÄÄRITYS

Tämän jälkeen kirjataan lokiin päivämäärä, kellonaika, tukiasema, salaustyyppi ja mahdollinen RADIUS-palvelin ja palataan odottamaan seuraavaa Go-napin painallusta.



Kuva 3: WLAN Verification Wizard -käyttöliittymä

Ohjelman alkutila on Kuva **3**:ssa oleva näkymä. Siitä näkee millaisia valintoja voidaan tehdä. Valitaan käytettävä radiotyyppi (normaali 802.11 b/g, 2,4GHz:n .11n tai 5GHz:n .11n), ilmasalauksen tyyppi (ei mitään, WEP, WPA/WPA2 –PSK, WPA/ WPA2 tai 802.1x), haluttu tukiasema (14 erilaista), mahdollinen EAP-tyyppi ja haluttu RADIUS -palvelin. Kaikki valinnat ovat dynaamisia eli tehty valinta sulkee pois mahdottomat valinnat. Esimerkkinä aloitustilanteessa olevan ”Open”-valinta, tällöin ilmarajapintaa ei salata mitenkään ja siten sertifikaattipohjaista tunnistusta ja RADIUS – palvelimia ei tarvita.

Käyttöliittymään on myös yhdistetty virranjakotukin käsiohjaus, jota voidaan käyttää myös muusta ohjelmasta riippumatta. Do Not Worry! – painike on virranjakotukin nollauspainike, mikäli on tarve nopeasti kytkeä virrat pois kaikista lähdöistä.

Lokitiedot kirjautuvat Done Setups – listaan ja jokaiselle riville voi myös tarvittaessa lisätä vapaamuotoista tekstiä. Save Setups – nappi tallentaa lokitiedot tekstitiedostoon ja Clear Setups – nappi tyhjentää lokin.

2.4. Ini-tiedostot

Ohjelman käynnistyessä perusasetukset haetaan VerificationWizard.ini – tiedostosta, joka sisältää tärkeimmät alkuvakiot. Ohessa esimerkki:

```
; WLAN VerificationWizard.ini
; Contains settings and variables
;
; Author: Jukka Issakainen, 2005 - 2009
;
[AP]
APSetupInfo = APSetupInfo.ini
[EAP]
;Types = TLS/PEAPv0, PEAPv1, TTLS-EAP-MSChapv2, TTLS-MSChapV2, LEAP, SIM, AKA
Types = TLS/PEAPv0, PEAPv1, TTLS, SIM, AKA
[Misc]
; these are just informational texts to show on screen, after AP-setup is completed
SSID = QAS_Verification
WEP64_Key = 1234567890, Key 2 (10 HEX digits)
WEP128_Key = 1234567890abcdef1234567890, Key 2 (26 HEX digits)
WPA(2)-PSK_Key = 12345678
TestUser = testi
TestUser_PWD = iopwlan
TestUser_Logon_Domain = WLAN-AUTH20

[RADIUS]
RadiusSetupInfo = RadiusSetupInfo.ini
```

Päädyin käyttämään ini-tiedostomuotoa, koska se oli helposti muokattavissa millä tahansa tekstieditorilla ja se voi sisältää kommentteja. Lisäksi voi määritellä avainsanan, jonka alla on vain kyseistä avainsanaa koskevia määrityksiä. Alun perin haaveilin käyttäväni vain yhtä määritystiedostoa, mutta pian kävi ilmi, että tarvittavan tiedon määrä on niin suuri, että editointi alkaa olla varsin hankalaa. Tämän johdosta päädyin käyttämään erillisiä tiedostoja tukiasemamäärityksille (esimerkissä APSetupInfo.ini,) ja tunnustuspalvelimille omaansa (esimerkissä RadiusSetupInfo.ini).

Tukiasemien alkutiedot luetaan käynnistyksen yhteydessä VerificationWizard.ini-tiedostossa määritellystä tiedostosta, tässä tapauksessa APSetupInfo.ini. Seuraavana lyhennetty esimerkki niistä:

```
; Access Point Info file
; Contains settings and variables
;
; Author: Jukka Issakainen, 2005 - 2009
;
; PDU_Port = Physical AC outlet # in APC Switched Rack PDU
; Model = Info abt AP model, not used, but clarifies ini-file
; Firmware = Info abt Firmware, not yet used
; IP =  IP-address of AP
; SSH = yes/no  If AP is capable of SSH
```

```
; 802.11n_2 = If AP supports 802.1n 2.4GHz mode, prefix for setupfile (e.g.
WNAP_N2.4_)
; 802.11n_5 = If AP supports 802.1n 5GHz mode, prefix for setupfile (e.g. WNAP_N5_
; UserID =   User name of AP administrative account
; Password = Password of AP administrative account
; GoRestore =  Path to config restore page inside Access Point, if exists
; RestoreFolder = Folder used to store config-files, empty folder, if Telnet/SSH is
used
; Open =  Open, no wep-key etc
; 802.1x= 802.1x with dynamic wep-key 128-bit and RADIUS
; WEP128 =   wep-key 128-bit, no RADIUS
; WPA-PSK = WiFi Protected Access, Pre Shared Key using TKIP + AES (a.k.a mixed WPA
SOHO-mode), no RADIUS
; WPA = WiFi Protected Access, RADIUS for EAP-types using TKIP + AES (a.k.a mixed WPA
Enterprise-mode)
; WPA2-PSK = WiFi Protected Access 2, Pre Shared Key using AES, no RADIUS
; WPA2 =  WiFi Protected Access 2, RADIUS for EAP-types using AES
; LEAP =  Cisco specific mode

[AP]
Models = A-Link WNAP, Apple Airport Extreme, Cisco 1231, Belkin N1 Vision, Buffalo
WZR-AG300NH, D-Link DIR-655, Linksys WRT54GS, Linksys WRT350N, Linksys WRT610N,
Netwjork 54Mbps, Thomson 585, Telewell TW-EA515, ZyXEL NBG-415N, ZyXel P-661HW-D1
RestoreModes = Open, WEP128, WPA-PSK, WPA2-PSK, WPA, WPA2, 802.1x
RestorePath = D:\Jukan\automation\AP_Setups
802Modes = 802.11 b/g, 802.11n (2.4GHz), 802.11n (5GHz)

[PDU]
; PDU_Enabled = Yes or No
; New interface after FW upgrade 2.70 or newer: <password><space>-c
PDU_Enabled = yes
PDU_IP = 10.10.32.17
PDU_User = wizard
PDU_Pwd = wizard -c

;############################ AP SETUP starts ########################################

[A-Link WNAP]
PDU_Port = 16
Model = WL524
Firmware = e2.04
IP = 10.10.32.151
SSH = no
802.11n_2 = WNAP_N2_
802.11n_5 = WNAP_N5_
UserID = admin
Password = admin
GoRestore = /saveconf.asp
RestoreFolder = \A-LinkWNAP
Open = A-Link_Open.dat
802.1x =
WEP64 =
WEP128 = A-Link_WEP128.dat
WPA-PSK = A-Link_wpa-psk_mixed.dat
WPA = A-Link_WPA_enterprise_mixed.dat
WPA2-PSK =
WPA2 = A-Link_WPA2_enterprise.dat
LEAP =
```

Ohjelman päänäkymässä oleva tukiasemalista luetaan käynnistyksen yhteydessä [AP]-lohkosta ja kullakin tukiasemalla on ruudulla näkyvää nimeä vastaava lohko tässä tiedostossa, esimerkiksi [A-LINK WNAP]. Tärkeimpinä tietoina tukiaseman IP-osoite, pääkäyttäjän nimi ja salasana ja kunkin kyseisessä tukiasemassa käytössäolevan salaus-

tavan sisältävä asetustiedosto. Asetustiedostojen salaukseen en katsonut olevan mitään tarvetta, koska ensisijassa ohjelmaani käytetään omassa laboratoriossani työnantajan tiloissa, jotka ovat varsin tarkkaan kulkusuojattuja. Toki, jos tilanne olisi niin vaatinut, olisi ollut mahdollista käyttää haluttua salausta näidenkin tietojen suojaamiseen.

Ylläoleva lista muodostaa myös tukiaseman ominaisuusmatriisin. Mikäli jossain suojausmuodossa ei lue tiedoston nimeä, kyseinen suojausmuoto ei ole käytettävissä kyseisellä tukiasemalla. Tätä tietoa käytetään hyväksi kun ohjelma sulkee pois käytöstä (tai mahdollistaa) ominaisuuksia (tukiasemia, salaustyyppejä tai RADIUS-palvelimia) tehtyjen valintojen mukaan. Mikäli jokin ominaisuus ei ole käytettävissä, muutetaan valintanappi harmaaksi, jottei käyttäjällä ole mahdollisuutta valita sitä ja näin saada aikaan mahdotonta tilannetta.

Esimerkkinä yllä WEP64 – ominaisuus ei ole käytössä tällä tukiasemalla. Kaikkia tukiasemien ominaisuuksia ei välttämättä käytetä testauksessa testauksen ajankäytön ja kattavuuden tasapainottamiseksi. Myöskään kaikki tukiasemat eivät sisällä kaikkia ominaisuuksia. Lisähuomiona voidaan mainita joidenkin tukiasemien hyödyntävän Telnet- tai SSH-yhteyttä. Tällöin ei tarvitse huolehtia erilaisten selainkäyttöliittymien omituisuuksista vaan asetukset voidaan tehdä suoraan Telnetillä/SSH:lla käyttäen kullekin tukiasemalle ominaisia käskyjä.

Aina kun valinta tehdään (802 -mode, Air encryption, Access Point, RADIUS Server) täytyy varmistaa valinnan järkevyys. Seuraavassa esimerkkinä koodin osa, jolla tukiaseman ominaisuudet luetaan kaksiuloitteiseen taulukkoon, jota käytetään pohjana valintojen tarkistuksessa:

2.5. Ominaisuusmatriisi

Tätä taulukkoa käytetään hyväksi myöhemmin ohjelmassa tehtäessä käyttäjän valintaan perustuvia toiminnallisuuksien poissulkemisia ja mahdollistamisia.

```
Dim $asApCapa[$iNumOfAP][$iNumOfAir + 2] ; +2 for 802.1n -modes
For $i = 1 To $asListAP[0]
   $asApCapa[$i][0] = $asListAP[$i] ; AP name for column #0
   For $j = 1 To $asListAir[0]
      $asApCapa[0][$j] = $asListAir[$j]
      $asApCapa[$i][$j] = StringStripWS(IniRead($sLocalPath &
      $sApIniFile, $asListAP[$i], $asListAir[$j], "NotFound"), 8)
   Next
   $asApCapa[0][$j] = ".1n prefix 2.4GHz"
   $asApCapa[$i][$j] = StringStripWS(IniRead($sLocalPath & $sApIniFile,
      $asListAP[$i], "802.1n_2", "NotFound"), 8)
                          ; 2.4 GHz 802.1n -mode prefix
```

```
   $asApCapa[0][$j + 1] = ".1n prefix 5GHz"
   $asApCapa[$i][$j + 1] = StringStripWS(IniRead($sLocalPath & $sApIniFile,
                                        $asListAP[$i], "802.1n_5", "NotFound"), 8)
                           ; 5 GHz 802.1n -mode prefix
Next
```

Ylläolevassa esimerkissä näkyy myös käyttämäni muuttujien nimeämistapa. AutoIt käyttää dollarimerkkiä kaikkien muuttujien edessä. Tämän lisäksi päätin kertoa muuttujan alussa minkä tyyppinen muuttuja on kyseessä: ”$asApCapa” tarkoittaa taulukkoa, joka muodostuu merkkijonoista (”$as” = Array string). ”ApCapa” taasen on lyhenne sanoista Access Point Capabilities eli tukiaseman ominaisuudet. ”$iNumOfAP” tarkoittaa muuttujaa, jonka tyyppi on kokonaisluku (”$i” = Integer) ja ”NumOfAP” tarkoittaa tukiasemien lukumäärää. AutoIt ei pakota tällaiseen merkitsemistapaan, vaan olen sen aikoinaan itselleni sopivaksi havainnut.

Kun tarvittavat valinnat on tehty, painetaan GO-nappia. Tällöin tarkistetaan mihin porttiin virtakiskossa on tukiasema liitetty ja kytketään virrat päälle.

3. TUKIASEMAN SÄÄTÄMINEN

Tukiaseman käynnistyminen vie oman aikansa ja ennenkuin se on täysin käynnistynyt, ei yhteyttä voi muodostaa. Tilannetta valvotaan lähettämällä ping-paketteja tukiaseman ip-osoitteeseen ja kun tukiasema vastaa pingiin, odotetaan vielä hetki ja vasta sitten käynnistetään selain ja otetaan yhteys tukiasemaan. Selaimeksi valittiin Internet Explorer 6, koska se tuli helposti serverin asennuksen yhteydessä. Toki mahdollista olisi ollut käyttää myös muita vaihtoehtoisia selaimia, mutta Internet Explorer tuli valittua. Lisäksi uusimmilla AutoIt:n versioilla oli mahdollista hyödyntää Windowsin luokkakirjastoja ja täten suoraan manipuloida web-sivun luokkakomponentteja objektitunnuksen avulla.

Selain käynnistetään ohjelman komentamana ja se ottaa yhteyden haluttuun tukiasemaan ja antaa osoitteeksi suoraan tukiaseman hallintasivun, joka on määritetty kullekin tukiasemalle erikseen GoRestore-parametrilla, joka löytyy VerificationWizard.ini - tiedoston parametrin ApSetupInfo määrittämästä tiedostosta, tässä tapauksessa ApSetupInfo.ini - tiedostosta. Erilaiset toimintomuodot ovat tallennettu kukin omaksi tiedostokseen, jonka nimi on määritetty kullekin tukiaseman ominaisuudelle erikseen. Tiedoston nimellä sinänsä ei ole mitään väliä, mutta selkeyden vuoksi päädyin käyttämään mahdollisimman kuvaavaa nimeä kuten”A_Link_WEP128.dat”. Tiedoston nimestä nä-

kee mistä tukiasemasta on kysymys sekä mikä on tukiaseman salausmuoto. Tässä tapauksessa kyseessä on A-Linkin tukiasema ja 128-bittinen WEP-salaus.

Tukiasemien käsittely on selkeyden vuoksi erotettu omaksi ohjelmakseen ”RestoreAP.exe”. Alun perin tukiasemien ohjaus oli pääohjelman sisällä, mutta tukiasemien määrän kasvaessa oli selkeämpää eriyttää se omaksi ohjelmakseen, jota kontrolloidaan pääohjelmasta. Lisäksi ohjelmiston kehityksen loppuvaiheessa mieleen tullut ”Replay”-ominaisuus olisi käytännöllisempää toteuttaa, mikäli tukiasemaohjaus olisi oma kokonaisuutensa. Ohjelman lähdekoodi löytyy liitteestä Liite 2: RestoreAP –ohjelman lähdekoodi.

Selaimen käynnistyttyä ja avattua tukiaseman varmuuskopiosivun ohjelma lähettää halutun määrän tabulaattorinäppäimen painalluksia, jotta päästään tekstisyöttökenttään. Sinne syötetään palautettavan tiedoston nimi polkuineen, siirrytään Ok-napille ja painetaan sitä. Tukiasema lataa halutun varmuuskopion ja käynnistyy uudelleen halutussa kokoonpanossa.

Uusimmissa AutoIt:n versioissa on mahdollista käsitellä suoraan tukiaseman websivun luokkakomponentteja. Tällöin selvitetään sopivalla apuohjelmalla halutun komponentin luokka ja yksikkötunnus (Instance), jolloin päästään suoraan manipuloimaan kyseistä komponenttia. Seuraavassa lyhyt esimerkki Linksys WRT54GS – tukiaseman käsittelystä, johon on yhdistetty molemmat toimintatavat:

3.1. Esimerkki: Linksys WRT54GS – tukiaseman ohjaus

```
Case $sSelectedAp = "LinksysWRT54GS" ;
  WinWaitActive("Connect")
  Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
  Sleep(2000)
  WinWait("Config Management")
  Sleep(2000)
  ControlClick("Config Management","","[CLASS:Internet Explorer_Server;
    INSTANCE:1]","left",1,373,300)
;Last 2 numbers are offset from top left in pixels for the center of the button
  Sleep(1000)
  Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
  Sleep(3000)
  Winwait("restore.cgi")
  Sleep(5000)
  ControlClick("restore.cgi","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,391,237)
  ; Succesful OK-btn, last 2 numbers are offset from top left in pixels for the center of the button
```

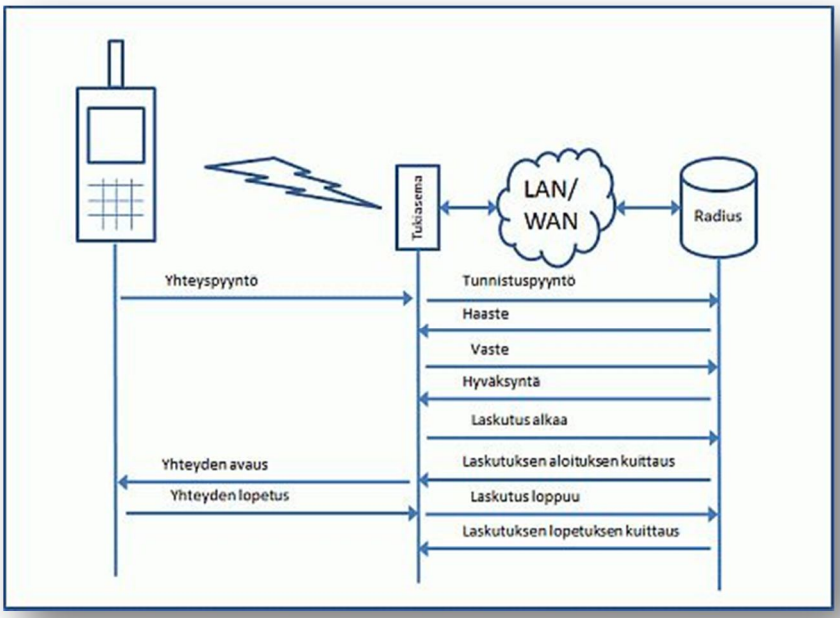
```
WinWait("Basic Setup")
Sleep(3000)
WinClose("Basic Setup")
```

Lopuksi vielä kirjoitetaan päiväys, kellonaika ja valittu konfiguraatio ohjelman pääikkunan alalaidassa olevaan kommentointikelpoiseen listaan (Done Setups), jotta pystyy myöhemmin tarkistamaan mitä kokoonpanoja on tullut käytettyä ja mitä omia kommentteja on tullut lisättyä. Nämä tiedot on myös mahdollista tallentaa tekstitiedostona painamalla Save List – nappia tai ohjelmasta poistuttaessa. Mikäli listaa ei ole tallennettu, ohjelman lopetuksen yhteydessä avataan tallennusikkuna ja tiedot voi tallentaa tai myös jättää tallentamatta. Koko ohjelmiston lähdekoodi on nähtävänä: Liite 1: WLAN Verfication Wizardin lähdekoodi

4. RADIUS-PALVELIMEN MÄÄRITYS

Kun käytetään yrityskäytön salauksia, tarvitaan useinmiten myös tunnistuspalvelinta. Tilannetta mutkistaa hiukan tosiasia, että ei ole olemassa yksittäistä palvelinta, joka osaa kaikki tarvittavat tunnistustavat. Testauksessani käytettävissä palvelimissa ajettiin Microsoftin Windows Server 2003/2008 käyttöjärjestelmiä. Tällöin luonteva valinta oli Microsoftin Internet Authentication Server (IAS), joka pystyy itse tekemään TLS- ja PEAPv0 - tunnistukset ja toimimaan niin sanottuna RADIUS-välityspalvelimena (proxy) muille servereille. PEAPv1 ja TTLS – tunnistuksia varten käytettiin Funkin (sittemin Juniper) ja Ciscon palvelimia. Kyseiset palvelimet olivat varsin yleisesti käytössä monissa yrityksissä. GSM- ja 3G- puhelinverkkojen sim-korttiin perustuvat EAP-SIM- ja -AKA-tunnistukset taas hoidettiin Nokian radiotestiverkon keskuksilla.

Seuraavana yksinkertaistettu periaatekuva RADIUS-tunnistuksesta:

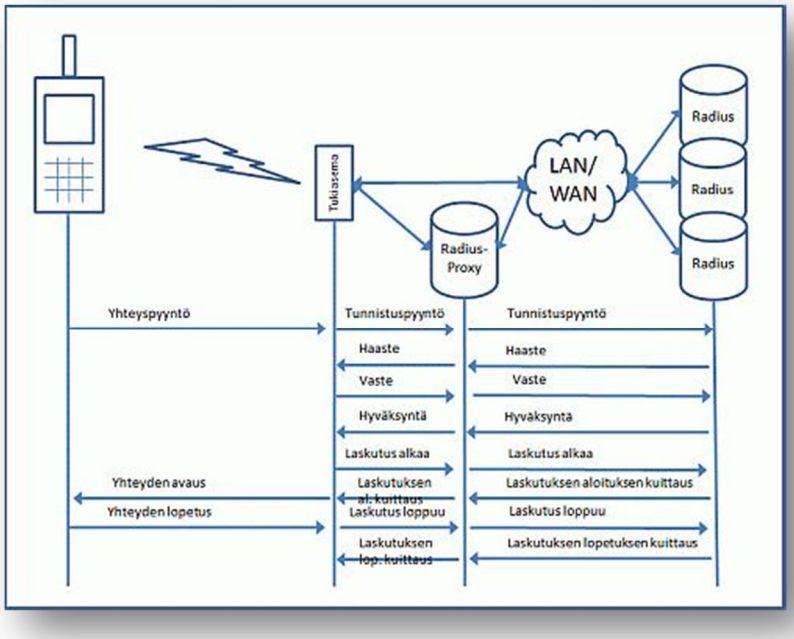


Kuva 4: RADIUS-tunnistuksen periaatekuva

RADIUS toimii asiakas-palvelin periaatteella ja tässä tapauksessa RADIUS-asiakkaana toimii tukiasema. Ylläkuvatussa yksinkertaistetussa signalointikuvassa on nähtävissä, että RADIUS-palvelin ei tunnistuksen jälkeen enää osallistu liikenteen käsittelyyn. Se antaa tukiasemalle tiedon onnistuneesta tunnistuksesta (tai hylkäyksestä), tukiasema lähettää RADIUS-palvelimelle laskutuksen aloituspyynnön ja tukiasema avaa yhteyden saatuaan RADIUS-palvelimelta kuittauksen laskutuksen aloituksesta. Tämän jälkeen liikennöinti sallitaan tukiasemasta eteenpäin. Yhteyden loputtua tukiasema antaa RADIUS-palvelimelle pyynnön laskutuksen lopetuksesta, jonka se vahvistaa.

Jos halutaan käyttää monia RADIUS-palvelimia esimerkiksi kuormantasauksen takia tai käytettäessä monia erilaisia tunnistusmenetelmiä, täytyy käyttää RADIUS-välityspalvelinta. Se ohjaa liikenteen halutulle RADIUS-palvelimelle.

Esimerkki RADIUS-välityspalvelimen toiminnasta:



Kuva 5: Periaatekuva RADIUS-välityspalvelimen toiminnasta

RADIUS-välityspalvelimelle määritetään mitä RADIUS-palvelinta käytetään kullekin tunnistustavalle.

Mikäli RADIUS-tunnistustapaa vaihdetaan toiseksi, voidaan helposti joutua tilanteeseen, että sillä hetkellä käytössä oleva RADIUS-palvelin ei mahdollista haluttua tunnistusta. Esimerkki: käytössä on Microsoft IAS, joka ymmärtää TLS- ja PEAPv0 - tunnistukset, mutta ei PEAPv1:tä. Jos halutaan tehdä PEAPv1 – tunnistus, tulee muuttaa käytössä oleva RADIUS-palvelin RADIUS-välityspalvelimeksi ja kertoa sille yhteydenottoparametrit toiseen palvelimeen, joka pystyy ymmärtämään PEAPv1-tunnistuksen. Ongelmaksi muodostui palvelimen asetusten muutto hallitusti. Palvelimen siirto välityspalvelimesta RADIUS-serveriksi ja takaisin tehdään yleensä pienellä ohjelmistolla, jota käytetään suoraan palvelimen konsolilta. Se on hieman kankea automatisoitavaksi ja vaatii operoinnin suoraan konsolilta. Pienen tuumailun jälkeen muistin lukeneeni Da-

niel Petrin loistavan artikkelin pienestä ohjelmasta nimeltä netsh (Configure TCP/IP from the Command Prompt 2009).

3.1. Netsh

Netsh on hieman tuntemattomampi ominaisuus Windowsissa (mukana jo Windows 2000:sta alkaen). Sillä voidaan tehdä monia asioita komentoriviltä, esimerkiksi muuttaa verkkomäärityksiä erittäin nopeasti. Netsh:lla on myös mahdollista tallentaa käytössä oleva verkkomääritys tiedostoksi ja tarvittaessa ladata se takaisin.

WLAN Verification Wizard – ohjelmisto käyttää tätä mahdollisuutta halutun RADIUS-palvelimen määritykseen ja käynnistää netsh:n halutun kokoonpanon latausta varten. Näin saadaan muutettua RADIUS-liikenteen ohjaus halutulle palvelimelle muutamassa sekunnissa.

Netsh edellyttää ohjelman ajon tapahtuvan paikallisella konsolilla, joten käytettäessä WLAN Verification Wizard -ohjelmaa joltain toiselta koneelta kuin RADIUS välityspalvelimelta on pystyttävä luomaan prosessitason etäyhteys kohdepalvelimeen. Tämä on mahdollista hoitaa esimerkiksi PCAnywherella, mutta se ei oikein sopinut tilanteeseen ja lisenssikin maksaa.

Onneksi ilmaisjakeluohjelmistoista löytyy WinInternals/Mark Russinovichin tekemä PsExec – ohjelma (Russinovich Mark PsExec 2008). Se on Telnetin kaltainen ohjelmisto, jolla voidaan ajaa prosesseja toisessa koneessa. PsExec:n avulla oli mahdollista suorittaa tarvittavien verkkomääritysten vaihto netsh:n avulla, aivan kuten olisin itse ollut kyseisen palvelimen konsolilla. Tämä mahdollisti työtiloissani olevien palvelimien siirron oikeaan konesaliin, jossa sähkön syöttö ja ilmastointi olivat kontrolloituja ja varmistettuja.

5. POHDINTA

Oli varsin mielenkiintoista kehittää työkalu selkeään tarpeeseen. Aikataulupaineista ja ohjelman työkaluluonteesta johtuen liikaa aikaa ei käytetty ulkonäön hiomiseen. Ohjelma sisältää pieniä loogisia virheitä, mutta niiden kanssa tulee helposti toimeen. Näiden viimeisten virheiden korjaus olisi todennäköisesti vaatinut liikaa aikaa saavutettuun hyötyyn nähden.

Jatkokehityssuunnitelmissa oli rakentaa ”Replay”-ominaisuus eli hyödyntää tallennettuja konfiguraatiolistauksia. Testattavaan laitteeseen oli jo tuloillaan oma ohjelmistonsa, jolla usb-liitynnän kautta voidaan saada laitteen tilatietoja ja lähettää komentoja laitteelle. Tällöin olisi ollut kohtuullisen yksinkertaista rakentaa täysautomatisoitu ympäristö testausta varten. Tämä ominaisuus jäi kuitenkin rakentamatta työtehtävien vaihtumisen vuoksi.

AutoIt osoittautui varsin kelvoksi kehitysympäristöksi ja kieleksi. Sillä on kohtuullisen helppoa tehdä nopeita prototyyppejä ja pieniä työkaluohjelmia olen sillä tämänkin jälkeen tehnyt pääasiassa omaan käyttöön.

Vaihtoehtoisia ratkaisuja mietittäessä olisi ollut täysin mahdollista tehdä samanlainen ohjelmisto esimerkiksi C#:lla tai Qt:lla, mutta se olisi vaatinut kohtuuttoman paljon aikaa kielten vaatiman lisäopettelun johdosta.

Ohjemaa käytettiin tuotantokäytössä usean vuoden ajan (2005 – 2009) ja viimeisin versio oli käytössä yli vuoden verran.

LÄHTEET

APC by Schneider Electric. Switched Rack PDU. Luettu 2006. http://www.apc.com/products/resource/include/techspec_index.cfm?base_sku=AP7950

Bennet, Jonathan, AutoIt Consulting Ltd. AutoIt: automation and scripting language. Luettu 2005. http://www.autoitscript.com/site/autoit/

Petri, Daniel. Configure TCP/IP from the Command Prompt. Luettu 2009. http://www.petri.co.il/configure_tcp_ip_from_cmd.htm

Russinovich, Mark. PsExcec. Tallennettu 2008. http://technet.microsoft.com/en-us/sysinternals/bb897553

Tatham, Simon. Using the command line connection tool Plink. Tallennettu 2006. http://the.earth.li/~sgtatham/putty/0.58/htmldoc/Chapter7.html

Wi-Fi Alliance. Organization. Luettu 1.9.2011. http://www.wi-fi.org/organization.php

Liite 1: WLAN Verfication Wizardin lähdekoodi

```
#
;===============================================================================
; Program Name:   VerificationWizard 802.1n -version
; Description:    Load pre-defined WLAN settings to Access Points and select RADIUS-server
;                 according to the user choices
; Parameter(s):   radius, ap, all: for debug; disables related function or both
; Requirement(s):
; Return Value(s):Selected AirEncryption, AccessPoint, EAP-type, RADIUS-server and possible
;                 WireShark / Ethereal-log
; Author(s):      Jukka Issakainen, Nokia TP/SP/CM/QA Services
;===============================================================================

#include <ButtonConstants.au3>
#include <GUIConstantsEx.au3>
#include <ListViewConstants.au3>
#include <StaticConstants.au3>
#include <WindowsConstants.au3>
#include <Array.au3>
#include <Constants.au3>

;#Include <WinAPI.au3> ; _WinAPI_SetSysColors($vElements, $vColors) for PDU buttons

Opt("GUIOnEventMode", 1)
Const $sIniFile = "VerificationWizard.ini"
$sVersion = "VerificationWizard: Restore AP & Radius as exec & new APC IF, 27.4.2009"
$sDefaultGrey = 0xD4D0C8
Const $sDefaultGreen = 0x00ff00
Const $iAllPortsOff = "all" ; address for all APC ports
Const $iNumOfPduPorts = 16
Global $iNumOfAP, $iNumOfAir, $iNumOfEAP, $iNumOfRAD, $sTextOfButton
Global $sSelected802, $sSelectedAir, $sSelectedAp, $sSelectedEap, $sSelectedRad, $sPrevious802,
$sPreviousAir, $sPreviousAp, $sPreviousEap, $sPreviousRad
Global $iCurrentLvItem, $iFirstLvItem, $tTextToSave, $iIsListSaved
Global $Port, $sRestoreFile, $iPlinkHandle
Global $sRadiusRestorePath, $sRadiusProxy, $sRadiusRestoreFile, $iRadiusSetStatus, $sErrorItem
$sOnColor = 0x00ff00
$sOffColor = $sDefaultGrey
$PduPrevState = "off"
$PduActionPhrase = ""
$iMsgBoxTimeOut = 3
$iWatchDog = 0
$iIsListSaved = 0 ; ListView not saved

#Region - Debug helpers
$DEBUG_List_Event_Arrays = 0 ; If = 1, txt-file will be written to @scriptdir & exit
$DEBUG_iAccessPointOff = 0
$DEBUG_iRadiusOff = 0
#EndRegion - Debug helpers

#Region - INI-file
$sLocalPath = @ScriptDir & "\" ; Add trailing backslash
$sApIniFile = IniRead($sLocalPath & $sIniFile, "AP", "APSetupInfo", "NotFound")
; Read the name of APinifile from Wizard.ini
$sRadiusIniFile = IniRead($sLocalPath & $sIniFile, "RADIUS", "RadiusSetupInfo", "NotFound"); Read
the name of Radiusinifile from Wizard.ini

#EndRegion - INI-file

#Region - Install helperfiles
FileInstall("psexec.exe", $sLocalPath)
#EndRegion - Install helperfiles
```

jatkuu

```
#Region - Read from INI-file

$asList802Mode = StringSplit(StringStripWS(IniRead($sLocalPath & $sApIniFile, "AP", "802Modes",
"NotFound"), 1), ",") ; 802.11 b/g, 802.11n (2.4GHz), 802.11n (5GHz)
$iNumOf802Mode = $asList802Mode[0] + 1

$asListAir = StringSplit(StringStripWS(IniRead($sLocalPath & $sApIniFile, "AP", "RestoreModes",
"NotFound"), 1), ",") ; Open, WEP128, WPA-PSK, WPA2-PSK, WPA, WPA2, 802.1x
$iNumOfAir = $asListAir[0] + 1

$asListAP = StringSplit(StringStripWS(IniRead($sLocalPath & $sApIniFile, "AP", "Models",
"NotFound"), 1), ","); Read from VerificationWizard.ini
$iNumOfAP = $asListAP[0] + 1;13+1 ; Read from INI-file
$sAPRestorePath = IniRead($sLocalPath & $sApIniFile, "AP", "RestorePath", "NotFound")
; Read from ApINI-file, not anymore VerificationWizard.ini
If StringRight($sAPRestorePath, 1) = "\" Then; check for trailing backslash in
   StringTrimRight($sAPRestorePath, 1)
; Remove trailing backslash, as it is used in APs RestoreFolder (ini-file)
EndIf

$sPDU_Enabled = StringUpper(StringStripWS(IniRead($sLocalPath & $sApIniFile, "PDU", "PDU_Enabled",
"NotFound"), 8)); is PDU around Yes/No
If $sPDU_Enabled = "YES" Then
   $sPDU_IP = StringStripWS(IniRead($sLocalPath & $sApIniFile, "PDU", "PDU_IP", "NotFound"), 8)
   $sPDU_User = IniRead($sLocalPath & $sApIniFile, "PDU", "PDU_User", "NotFound")
   $sPDU_Pwd = IniRead($sLocalPath & $sApIniFile, "PDU", "PDU_Pwd", "NotFound")
EndIf

$iPDU_Previous = 0 ; Initial value for PDU Port

#Region - Define APC PDU
Dim $aAPs_and_Ports[$iNumOfPduPorts + 1][4]
Dim $aUsedPduPorts[$iNumOfPduPorts + 1]; just in case, mark all used ports to this table
For $i = 1 to $asListAP[0]
      $iPdu_Port = StringStripWS(IniRead($sLocalPath & $sApIniFile, $asListAP[$i], "PDU_Port",
"NotFound"), 1)
   $aAPs_and_Ports[$iPdu_Port][0] = $asListAP[$i]
   $aAPs_and_Ports[$iPdu_Port][1] = $iPdu_Port
   $aAPs_and_Ports[$iPdu_Port][3] = "off" ; Default Port State
   ;$sAPs_and_Ports = $sAPs_and_Ports & @CRLF & $aAPs_and_Ports[$i][0] & " Port = " &
$aAPs_and_Ports[$i][1]
Next
#EndRegion - Define APC PDU

#Region - Read Access Point capabilities from INI-file, if a file name is available = AP is capable
of that mode, $asApCapa[][]
Dim $asApCapa[$iNumOfAP][$iNumOfAir + 2] ; +2 for 802.1n -modes
For $i = 1 To $asListAP[0]
   $asApCapa[$i][0] = $asListAP[$i] ; AP name for column #0
   For $j = 1 To $asListAir[0]
      $asApCapa[0][$j] = $asListAir[$j]
      $asApCapa[$i][$j] = StringStripWS(IniRead($sLocalPath & $sApIniFile, $asListAP[$i],
$asListAir[$j], "NotFound"), 8)
   Next
   $asApCapa[0][$j] = ".1n prefix 2.4GHz"
   $asApCapa[$i][$j] = StringStripWS(IniRead($sLocalPath & $sApIniFile, $asListAP[$i], "802.1n_2",
"NotFound"), 8) ; 2.4 GHz 802.1n -mode prefix
   $asApCapa[0][$j + 1] = ".1n prefix 5GHz"
   $asApCapa[$i][$j + 1] = StringStripWS(IniRead($sLocalPath & $sApIniFile, $asListAP[$i],
"802.1n_5", "NotFound"), 8) ; 5 GHz 802.1n -mode prefix
Next
#EndRegion - Read Access Point capabilities from INI-file, if a file name is available = AP is
capable of that mode, $asApCapa[][]
```

```
$asListEap = StringSplit(StringStripWS(IniRead($sLocalPath & $sIniFile, "EAP", "Types", "NotFound"),
8), ","); Read from VerificationWizard.ini
$iNumOfEAP = $asListEap[0] + 1 ;$iNumOfEAP = 5 ; tls/peapv0, ttls, peapv1, sim, aka

$asListRADIUS = StringSplit(StringStripWS(IniRead($sLocalPath & $sRadiusIniFile, "RADIUS",
"Servers", "NotFound"), 1), ","); Read from VerificationWizard.ini
$iNumOfRAD = $asListRADIUS[0] + 1

#Region - Read RADIUS server capabilities from INI-file, $asRadCapa[][]
Dim $asRadCapa[$iNumOfRAD][$iNumOfEAP]
For $i = 1 To $asListRADIUS[0]
   $asRadCapa[$i][0] = $asListRADIUS[$i] ; name of Radius server for column #0
   For $j = 1 To $asListEap[0]
      $asRadCapa[0][$j] = $asListEap[$j] ; insert EAP-type to top column, easier to read
      $asRadCapa[$i][$j] = StringStripWS(IniRead($sLocalPath & $sRadiusIniFile, $asListRADIUS[$i],
$asListEap[$j], "NotFound"), 8) ; Read server's support for EAP-types from RadiusSetupInfo.ini
   Next
Next

#EndRegion - Read RADIUS server capabilities from INI-file, $asRadCapa[][]

#EndRegion - Read from INI-file

#Region - Screen component variables

$iRadioGrpTop = 10
$iRadioGrpWidth = 140
$iGrpSeparator = 20
$iRadioBtnSeparator = 25

$iRadioBtnTop = $iRadioGrpTop + $iGrpSeparator ; 33
$iRadioBtnLeftFromGrp = 10
$iRadioBtnHeight = $iGrpSeparator;17
$iRadioTxtWidth = $iRadioGrpWidth - 2 * $iRadioBtnLeftFromGrp;130

$i802GrpTop = $iRadioGrpTop ; New group above Air
$i802GrpLeft = 10
$i802GrpHeight = $asList802Mode[0] * $iRadioBtnSeparator + $iRadioBtnTop
$i802Left = $i802GrpLeft + $iRadioBtnLeftFromGrp

$iAirGrpTop = $iRadioGrpTop + $i802GrpHeight + $iGrpSeparator
$iAirGrpLeft = $i802GrpLeft ; 8 !New group above Air
$iAirGrpHeight = $asListAir[0] * $iRadioBtnSeparator + $iRadioBtnTop
$iAirLeft = $iAirGrpLeft + $iRadioBtnLeftFromGrp ; 8+10

$iApGrpTop = $iRadioGrpTop
$iApGrpLeft = $i802GrpLeft + $iRadioGrpWidth + $iGrpSeparator ; 8+145+20= 173
$iApGrpHeight = $asListAP[0] * $iRadioBtnSeparator + $iRadioBtnTop
$iApLeft = $iApGrpLeft + $iRadioBtnLeftFromGrp;166 173+10 = 183

$iEapGrpTop = $iRadioGrpTop
$iEapGrpLeft = $iApGrpLeft + $iRadioGrpWidth + $iGrpSeparator ; 173+145+20=338
$iEapGrpHeight = $asListEap[0] * $iRadioBtnSeparator + $iRadioBtnTop
$iEapLeft = $iEapGrpLeft + $iRadioBtnLeftFromGrp;342 338+10=348

$iRadGrpTop = $iRadioGrpTop
$iRadGrpLeft = $iEapGrpLeft + $iRadioGrpWidth + $iGrpSeparator ; 338+145+20=503 $asListRADIUS
$iRadGrpHeight = $asListRADIUS[0] * $iRadioBtnSeparator + $iRadioBtnTop
$iRadLeft = $iRadGrpLeft + $iRadioBtnLeftFromGrp;510 503+10=513

$iListViewHeight = 132
$iListViewWidth = 615
```

```
If $asListAP[0] > 12 Then
   $iMainHeight = $iRadioGrpTop + $iGrpSeparator + $iNumOfAP * $iRadioBtnSeparator + $iGrpSeparator
+ $iListViewHeight + $iGrpSeparator
Else
   $iMainHeight = 580
EndIf
$iMainWidth = 790

$iListViewTop = $iMainHeight - $iRadioGrpTop - $iGrpSeparator - $iListViewHeight - $iGrpSeparator
#EndRegion - Screen component variables

Global $ai802Event[$iNumOf802Mode], $aiAirEvent[$iNumOfAir], $aiApEvent[$iNumOfAP],
$aiEapEvent[$iNumOfEAP], $aiRadEvent[$iNumOfRAD], $aiPowerEvent[$iNumOfPduPorts + 1]

#Region ### START Koda GUI section ### Form=c:\data\automation\802.1n-
versio\wizard_on_event_mode\main.kxf

$Main = GUICreate($sVersion , $iMainWidth, $iMainHeight, 97, 45, BitOR($WS_OVERLAPPEDWINDOW,
$WS_CLIPSIBLINGS)) ; 817, 587, 97, 45
GUISetOnEvent($GUI_EVENT_CLOSE, "MainClose")
GUISetOnEvent($GUI_EVENT_MINIMIZE, "MainMinimize")
GUISetOnEvent($GUI_EVENT_MAXIMIZE, "MainMaximize")
GUISetOnEvent($GUI_EVENT_RESTORE, "MainRestore")
;GUICtrlCreateGroup("", -99, -99, 1, 1)

#Region - Create 802 Group Buttons
$Group_802 = GUICtrlCreateGroup(" 802-Mode ", $i802GrpLeft, $i802GrpTop, $iRadioGrpWidth,
$i802GrpHeight)
For $i = 1 To $asList802Mode[0]
   $ai802Event[$i] = GUICtrlCreateRadio($asList802Mode[$i], $i802Left, $iGrpSeparator + $i802GrpTop
+ ($i - 1) * $iRadioBtnSeparator, $iRadioTxtWidth, $iRadioBtnHeight)
   GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
   GUICtrlSetOnEvent(-1, "_NButton") ; default value
Next
GUICtrlSetState($ai802Event[1], $GUI_CHECKED) ; 1st selection as default
$sSelected802 = 1
$sPrevious802 = 1
GUICtrlCreateGroup("", -99, -99, 1, 1)
#EndRegion - Create 802 Group Buttons

#Region - Create Air Encryption Buttons

$Group_Air = GUICtrlCreateGroup(" Air Encryption ", $iAirGrpLeft, $iAirGrpTop, $iRadioGrpWidth,
$iAirGrpHeight);209)
For $i = 1 To $asListAir[0]
   $aiAirEvent[$i] = GUICtrlCreateRadio($asListAir[$i], $iAirLeft, $iGrpSeparator + $iAirGrpTop +
($i - 1) * $iRadioBtnSeparator, $iRadioTxtWidth, $iRadioBtnHeight)
   GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
   ; koe
   ;GUICtrlSetOnEvent(-1, $asListAirClick[$i])
   GUICtrlSetOnEvent(-1, "_AirButton")
Next
GUICtrlSetState($aiAirEvent[1], $GUI_CHECKED); 1st selection as default
$sSelectedAir = 1
$sPreviousAir = 1
GUICtrlCreateGroup("", -99, -99, 1, 1)
;_ArrayDisplay($aiAirEvent)
;_ArrayDisplay($asListAir)

#EndRegion - Create Air Encryption Buttons

#Region - Create AP Buttons
```

```
$Group_AP = GUICtrlCreateGroup(" Access Point ", $iApGrpLeft, $iApGrpTop, $iRadioGrpWidth,
$iApGrpHeight) ; 152, 8, 153, 369
For $i = 1 To $asListAP[0];$iNumOfAP -1
   $aiApEvent[$i] = GUICtrlCreateRadio($asListAP[$i], $iApLeft, $iRadioBtnTop + ($i - 1) *
$iRadioBtnSeparator, $iRadioTxtWidth, $iRadioBtnHeight)
   GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
   ;GUICtrlSetOnEvent(-1, $aApClick[$i])
   GUICtrlSetOnEvent(-1, "_ApButton")
Next
GUICtrlSetState($aiApEvent[1], $GUI_CHECKED); 1st selection as default
$sSelectedAp = 1
$sPreviousAp = 1
GUICtrlCreateGroup("", -99, -99, 1, 1)
;_ArrayDisplay($aiApEvent)
;_ArrayDisplay($asListAP)
#EndRegion - Create AP Buttons

#Region - Create EAP Buttons
$Group_EAP = GUICtrlCreateGroup(" EAP-Type", $iEapGrpLeft, $iEapGrpTop, $iRadioGrpWidth,
$iEapGrpHeight) ; 328, 8, 145, 185
For $i = 1 To $asListEap[0]
   $aiEapEvent[$i] = GUICtrlCreateRadio($asListEap[$i], $iEapLeft, $iRadioBtnTop + ($i - 1) *
$iRadioBtnSeparator, $iRadioTxtWidth, $iRadioBtnHeight)
   GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
   GUICtrlSetOnEvent(-1, "_EapButton")
   GUICtrlSetState(-1, $GUI_DISABLE); Startup situation
Next
GUICtrlSetState($aiEapEvent[1], $GUI_CHECKED); 1st selection as default
$sSelectedEap = 1
$sPreviousEap = 1
GUICtrlCreateGroup("", -99, -99, 1, 1)
;_ArrayDisplay($aiEapEvent)
;_ArrayDisplay($asListEap)
#EndRegion - Create EAP Buttons

#Region - Create RADIUS Buttons
$Group_RADIUS = GUICtrlCreateGroup(" RADIUS server ", $iRadGrpLeft, $iRadGrpTop, $iRadioGrpWidth,
$iRadGrpHeight) ; 496, 8, 145, 185
For $i = 1 To $asListRADIUS[0]
   $aiRadEvent[$i] = GUICtrlCreateRadio($asListRADIUS[$i], $iRadLeft, $iRadioBtnTop + ($i - 1) *
$iRadioBtnSeparator, $iRadioTxtWidth, $iRadioBtnHeight)
   GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
   GUICtrlSetOnEvent(-1, "_RadiusButton")
   GUICtrlSetState(-1, $GUI_DISABLE); Startup situation
Next
GUICtrlSetState($aiRadEvent[1], $GUI_CHECKED); 1st selection as default
$sSelectedRad = 1
$sPreviousRad = 1
GUICtrlCreateGroup("", -99, -99, 1, 1)
;_ArrayDisplay($aiRadEvent)
;_ArrayDisplay($asListRADIUS)
#EndRegion - Create RADIUS Buttons

#Region - Create Go/Exit
$iGoBtnLeft = $iRadGrpLeft + $iRadioGrpWidth + $iGrpSeparator
$iGoBtnTop = $iRadioGrpTop + $iGrpSeparator - 10
$iGoHeight = 3 * $iGrpSeparator ; 60
$iGoWidth = 6 * $iGrpSeparator ; 120

$ButtonGo = GUICtrlCreateButton("Go!", $iGoBtnLeft, $iGoBtnTop, $iGoWidth, $iGoHeight, 0) ; 664, 24,
120, 73, 0 left, top, width, height
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "_ButtonGoClick")
```

```
$ButtonExit = GUICtrlCreateButton("Exit", $iGoBtnLeft, $iGoBtnTop + $iGoHeight + $iGrpSeparator,
$iGoWidth, $iGoHeight, 0) ; 664, 112, 120, 73, 0
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "MainClose")
#EndRegion - Create Go/Exit

#Region - Create Setups ListView & Buttons

$Group_Setups = GUICtrlCreateGroup("Done Setups", $i802GrpLeft, $iListViewTop, $iMainWidth - 2 *
$iRadioGrpTop, $iListViewHeight + 2 * $iGrpSeparator - 5) ; 16, 392, 785, 169 left top width height
$ListView1 = GUICtrlCreateListView("Time | Date | 802-mode | Air | Access Point | EAP-type |
Radius", $iAirGrpLeft + $iRadioBtnLeftFromGrp, $iListViewTop + $iGrpSeparator, $iListViewWidth,
$iListViewHeight) ; 32, 416, 601, 132
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "ListView1Click")
$Btn_Save_List = GUICtrlCreateButton("Save Setups", $iGoBtnLeft, $iListViewTop + $iGrpSeparator,
$iGoWidth, $iGoHeight, 0) ; 666, 422, 120, 49, 0
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "Btn_Save_ListClick")
$Btn_Clear_List = GUICtrlCreateButton("Clear Setups", $iGoBtnLeft, $iListViewTop + $iGrpSeparator +
$iGoHeight + $iGrpSeparator - 10, $iGoWidth, $iGoHeight, 0) ; 666, 492, 120, 49, 0
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "_ClearListView")
GUICtrlCreateGroup("", -99, -99, 1, 1)
#EndRegion - Create Setups ListView & Buttons

;#cs
#Region - Create Power Control Buttons
$k = 0
$iPwrBtnTop = 232
$iPwrBtnWidth = 73
$iPwrBtnHeight = 33
$iApcGrpWidth = 450 ;
$iApcGrpHeight = 169
$iSeparator = $iGrpSeparator / 2 ; default 20
$iButtonsInRow = 4
$iButtonRows = 4
$PduPrevState = ""
$PduActionPhrase = ""
$iButtonWidth = ($iApcGrpWidth - 2 * $iSeparator) / $iButtonsInRow; 110
$iButtonHeight = ($iApcGrpHeight - 3 * $iSeparator - $iApcGrpHeight / 4) / $iButtonsInRow; 30
$iPanicButtonWidth = $iApcGrpWidth - 2 * $iSeparator; $iButtonRows * $iButtonWidth
$iPanicButtonHeight = 2 * $iButtonHeight
$sfont = "Ariel"
$iDefaultFontSize = $iPwrBtnWidth / 11

$_AP_Control = GUICtrlCreateGroup(" Access Point Power Control", 328, 208, $iApcGrpWidth,
$iApcGrpHeight); 328, 208, 473, 169
$APC_Btn_AllOff = GUICtrlCreateButton("Do Not Worry!", $iGoBtnLeft, 232, 120, 130, 0) ; 664, 232,
120, 129, 0 name, left, top, width, height
GUICtrlSetResizing(-1, $GUI_DOCKAUTO) ; Resize automatically according to window size
GUICtrlSetOnEvent(-1, "_AllPortsOff")
; $aAPs_and_Ports[$i][0] = AP_Name, $aAPs_and_Ports[$i][1] = Port, $aAPs_and_Ports[$i][2] = Button
@GUI_CTRLID; $aAPs_and_Ports[$i][3] = Button State; NOTE APC PDU has 16 ports
For $i = 1 To $iButtonsInRow ; PduBtnCtrlId goes to $aAPs_and_Ports[$i][2]
   For $j = 1 To $iButtonRows
      $aUsedPduPorts[$k + $j] = GUICtrlCreateButton("Free, Port # ", $iEapLeft + ($j - 1) *
$iPwrBtnWidth, $iPwrBtnTop + ($i - 1) * $iPwrBtnHeight, $iPwrBtnWidth, $iPwrBtnHeight,
$BS_MULTILINE)
      GUICtrlSetResizing(-1, $GUI_DOCKAUTO)
; Resize automatically according to window size
      GUICtrlSetFont(-1, $iDefaultFontSize, 700, 1, $sfont)
      GUICtrlSetOnEvent(-1, "_PduButton")
```

```
   Next
   $k = $k + 4
Next
For $i = 1 To $iNumOfPduPorts
   If $aAPs_and_Ports[$i][0] = "" Then
      GUICtrlSetState($aUsedPduPorts[$i],$GUI_DISABLE)
   Else
      $aAPs_and_Ports[$i][2] = $aUsedPduPorts[$i]
      GUICtrlSetData($aAPs_and_Ports[$i][2], $aAPs_and_Ports[$i][0])
; Put the correct name to correct port#
   EndIf
Next
;_ArrayDisplay($aAPs_and_Ports)
GUICtrlCreateGroup("", -99, -99, 1, 1)
#EndRegion - Create Power Control Buttons

If $DEBUG_List_Event_Arrays Then ; create a file of all event arrays
   _Debug_Events()
EndIf
$iFirstLvItem = 0 ; reset ListView items

GUISetState(@SW_SHOW)
#EndRegion ### END Koda GUI section ###

_APC_GetState()
While 1
   Sleep(1000)
   $iWatchDog = $iWatchDog + 1
   If $iWatchDog = 60 Then ; Simple timer just to update PDU button states
      $iWatchDog = 0
      _APC_GetState()
   EndIf
WEnd

#Region - N Button Clicked

Func _NButton()
   $sPrevious802 = $sSelected802
   $sSelected802 = @GUI_CtrlId ; if selected >4 then we are in .1n -mode
   _ToggleApState(@GUI_CtrlId) ; if n-mode -> dim all but n-aps, else enable all, $asApCapa

EndFunc   ;==>_NButton
#EndRegion - N Button Clicked

#Region - Air Encryption Button Clicked - functions
Func _AirButton()
   $sPreviousAir = $sSelectedAir
   $sSelectedAir = @GUI_CtrlId
   $iIndex = _ArraySearch($aiAirEvent, $sSelectedAir)

   If $iIndex > 4 Then ; Enterprise authentication, enable EAP & Radius
      _ToggleEapState($GUI_ENABLE)
      _ToggleRadiusState($GUI_ENABLE)
      ;If $iIndex = 7 Then ; 802.1x
      ;_ToggleApState($iIndex) ; send
      ;EndIf
   Else ; Personal authentication, disable EAP & Radius
      _ToggleEapState($GUI_DISABLE)
      _ToggleRadiusState($GUI_DISABLE)
   EndIf
   _ToggleApState(@GUI_CtrlId) ;

EndFunc   ;==>_AirButton
```

```
#EndRegion - Air Encryption Button Clicked - functions

#Region - Access Point Clicked -functions
Func _ApButton()
   $sSelectedAp = @GUI_CtrlId
EndFunc   ;==>_ApButton

#EndRegion - Access Point Clicked -functions

#Region - EAP Button Clicked - functions
Func _EapButton()
   $sSelectedEap = @GUI_CtrlId
   _ToggleRadiusState($GUI_ENABLE)
EndFunc   ;==>_EapButton

#EndRegion - EAP Button Clicked - functions

#Region - RADIUS Button Clicked - functions
Func _RadiusButton()
   $sSelectedRad = @GUI_CtrlId
   _ToggleEapState($GUI_ENABLE)
EndFunc   ;==>_RadiusButton
#EndRegion - RADIUS Button Clicked - functions

#Region - APC Button Clicked - functions

Func _AllPortsOff()
   For $i = 1 to $iNumOfPduPorts
      GUICtrlSetBkColor($aAPs_and_Ports[$i][2],$sOffColor)    ; Grey
      $aAPs_and_Ports[$i][3] = "off"
   Next
   _TogglePDU($iAllPortsOff, "off") ; Direct cut to the source... :-)
EndFunc ; ==> _AllPortsOff()

Func _PduButton()
   _TogglePduBtnState(@GUI_CtrlId)
EndFunc   ;==>_PduButton

Func _ConnectToAPC() ; Return $iPlinkHandle if connect succesful, 0 if NOK
   $IsPduOnLine = Ping($sPDU_IP, 500); wait 500ms
   If $IsPduOnLine = 0 Or @error Then ; = "Off" Then
      $sPDU_Enabled = "No"
      Return 0
   Else
      $sPDU_Enabled = "YES"
      $iPlinkHandle = Run(@ComSpec & ' /c ' & $sLocalPath & 'plink.exe -telnet ' & $sPDU_IP, '',
@SW_HIDE, 7);$STDERR_CHILD + $STDOUT_CHILD) ;
      Sleep(2000)
      While Not ProcessExists($iPlinkHandle)
; just in case someone has logged in to cmd-interface at the same moment
         MsgBox(48, "Communication Error to APC PDU", "Command Line Interface is reserved for
someone else..." & @CRLF & @CRLF & "Will try again after 10 secs...", $iMsgBoxTimeOut)
         Sleep(10000)
         $iPlinkHandle = Run(@ComSpec & ' /c ' & $sLocalPath & 'plink.exe -telnet ' & $sPDU_IP, '',
@SW_HIDE, 7);$STDERR_CHILD + $STDOUT_CHILD) ;
         Sleep(2000)
      WEnd
      While 1 ;waits for "User"
         $text = StdoutRead($iPlinkHandle)
         $oktogo = StringRegExp($text, ".*User*")
         If $oktogo = 1 Then ExitLoop
      WEnd
```

```
      StdinWrite($iPlinkHandle, $sPDU_User & @CR)
      While 1 ;waits for "password"
         $text = StdoutRead($iPlinkHandle)
         $oktogo = StringRegExp($text, ".*Password.*")
         If $oktogo = 1 Then ExitLoop
      WEnd
      StdinWrite($iPlinkHandle, $sPDU_Pwd & @CR)
      Sleep(500)
      Return $iPlinkHandle
   EndIf

EndFunc   ;==>_ConnectToAPC()


Func _TogglePduBtnState($iCtrlId)
   ; check the state of pressed button and change it
   ; initially try to read from PDU the start up state or if not too time consuming every time
button is pressed...
   ; @GUI_CTRLID replaced with $iCtrlId
   ; $aAPs_and_Ports[$i][0] = AP_Name, $aAPs_and_Ports[$i][1] = Port, $aAPs_and_Ports[$i][2] =
Button @GUI_CTRLID; $aAPs_and_Ports[$i][3] = Button State
   For $i = 1 To $iNumOfPduPorts
      If $iCtrlId = $aAPs_and_Ports[$i][2] Then ; $iCtrlId = $aAPs_and_Ports[$i][2]
         If $aAPs_and_Ports[$i][3] = "Off" Then
            GUICtrlSetBkColor($iCtrlId, $sOnColor) ; Green
            $aAPs_and_Ports[$i][3] = "On"
         Else
            GUICtrlSetBkColor($iCtrlId, $sOffColor) ; Grey
            $aAPs_and_Ports[$i][3] = "Off"
         EndIf
         _TogglePdu($aAPs_and_Ports[$i][1], $aAPs_and_Ports[$i][3])
      EndIf
   Next
EndFunc   ;==>_TogglePduBtnState

Func _TogglePDU($Port, $PduAction)

   If _ConnectToAPC() = 0 Then   ; = "Off" Then
      MsgBox(0, "TogglePDU", "$PduPrevState = " & $PduPrevState & @CRLF & "$PduAction = " &
$PduAction & @CRLF & "$Port = " & $Port & @CRLF & "PDU error = " & @error,$iMsgBoxTimeOut)
   Else

      StdinWrite($iPlinkHandle,$PduAction & " " & $Port & @CR) ; e.g. on 10, off 10, off all
      Sleep(500)
      While 1 ;waits for "APC>" confirmation
         $text = StdoutRead($iPlinkHandle)
         $oktogo = StringRegExp($text,".*APC>*")
         If $oktogo = 1 Then ExitLoop
      Wend
      sleep(500)
      StdinWrite($iPlinkHandle,"Bye" & @CR) ; 4- Logout
   EndIf
   Sleep(500)
   ProcessClose("PLINK.EXE") ; just in case Plink gets to hang
   $PduPrevState = $PduAction
EndFunc ; ==> Func _TogglePDU($iPort, $PduAction)

Func _APC_GetState() ; Read port states from APC, gets called from main loop and after GO
   If _ConnectToAPC() = 0 Then   ; = "Off" Then
      MsgBox(0, "APC_GetState", "PDU Off-line", $iMsgBoxTimeOut)
   Else
      Sleep(500)
      ;#cs
```

```
      While 1 ; After this loop $line contains what we want
         $line = StdoutRead($iPlinkHandle)
         $oktogo = StringRegExp($line,".*>*")
         If $oktogo = 1 Then ExitLoop
      Wend
      ;#ce
;MsgBox(0, "Lopputulos:", $line)
      StdinWrite($iPlinkHandle,"bye" & @CR)
      Sleep(500)

      $APC_Out = StringSplit($line, @CR, 0)
;_ArrayDisplay($APC_Out, "APC_OUT Arrayna")
      $k = 0
      While 1 ; After this loop $line contains what we want
         $oktogo = StringRegExp($APC_Out[$k],"1:")
         If $oktogo = 1 Then ExitLoop
; now we know from which index the actual data begins, should be [10] with the new version of APC fw
         $k = $k +1
      Wend
;MsgBox(0, "$k", $k)
; $aAPs_and_Ports[$i][0] = AP_Name, $aAPs_and_Ports[$i][1] = Port, $aAPs_and_Ports[$i][2] = Button
@GUI_CTRLID; $aAPs_and_Ports[$i][3] = Button State; NOTE APC PDU has 16 ports
      For $l = $k To $APC_Out[0] -2; If String contains "N" as 9th char from left-> Port is ON

         If StringInStr(StringLeft($APC_Out[$l],10), "ON") Then ; $l -10 contains the port number
            GUICtrlSetBkColor($aAPs_and_Ports[$l - $k +1][2],$sOnColor)    ; Green
            $aAPs_and_Ports[$l - $k +1][3] = "On"
         Else
            GUICtrlSetBkColor($aAPs_and_Ports[$l - $k +1][2],$sOffColor)    ; Grey
            $aAPs_and_Ports[$l - $k +1][3] = "Off"
         EndIf

      Next
   EndIf ; ==> $IsPduOnLine
;_ArrayDisplay($APC_Out, "Porttien tila")
EndFunc ; ==>_APC_GetState()

Func _PDU_Control($Port, $PduAction); Port# On/Off as parameters

   $PduPrevState = $PduAction
   If $PduAction = "On" Then
      ;$PduAction = 1 ; 1 = Immediate On
      $PduActionPhrase = "Immediate On"
   Else
      ;$PduAction = 2 ; Immediate Off
      $PduActionPhrase = "Immediate Off"
   EndIf
   $iPlinkHandle = _ConnectToAPC()
   If $iPlinkHandle Then
      StdinWrite($iPlinkHandle, $PduAction & " " & $Port & @CR) ; 1- Immediate On, 2- Immediate Off
      Sleep(500)
      StdinWrite($iPlinkHandle, "Bye" & @CR) ; 4- Logout
   EndIf
EndFunc   ;==>_PDU_Control

#EndRegion - APC Button Clicked - functions

#Region - Main Window Button Clicked - functions
;Func Btn_Clear_ListClick()

;EndFunc
Func Btn_Save_ListClick() ; format: time | date | 802-mode | Air | AP | EAP | Radius @CRLF
   $tTextToSave = ""
```

```
   $sSaveFile = "Done Setups.txt"
   For $i = 1 To $iCurrentLvItem - $iFirstLvItem + 1
      $tTextToSave = $tTextToSave & @CRLF & StringTrimRight(GUICtrlRead($iFirstLvItem + $i - 1), 1)
      $tTextToSave = StringReplace($tTextToSave, "|", ",", 0); create comma-delimed list
   Next
   $iIsListSaved = _SaveTxt($tTextToSave, $sSaveFile) ; returns 0 or 1

EndFunc   ;==>Btn_Save_ListClick

Func _ButtonGoClick()
   For $i = 1 To $asList802Mode[0]
      If GUICtrlGetState($ai802Event[$i]) = $GUI_ENABLE + $GUI_SHOW And GUICtrlRead($ai802Event[$i])
= $GUI_CHECKED Then
         $sSelected802 = $asList802Mode[$i]
      EndIf
   Next
   For $i = 1 To $asListAir[0]
      If GUICtrlGetState($aiAirEvent[$i]) = $GUI_ENABLE + $GUI_SHOW And GUICtrlRead($aiAirEvent[$i])
= $GUI_CHECKED Then
         $sSelectedAir = $asListAir[$i]
      EndIf
   Next
   For $i = 1 To $asListAP[0]
      If GUICtrlGetState($aiApEvent[$i]) = $GUI_ENABLE + $GUI_SHOW And GUICtrlRead($aiApEvent[$i]) =
$GUI_CHECKED Then
         $sSelectedAp = $asListAP[$i]
      EndIf
   Next
#cs
   For $i = 1 To $iNumOfPduPorts ; select correct APs Power button and PDU port, 3.4.2009
      If $sSelectedAp = $aAPs_and_Ports[$i][0] Then
         $iCtrlId = $aAPs_and_Ports[$i][2]
         $iPDU_Port = $aAPs_and_Ports[$i][1]
      EndIf
   Next
#ce
   $sSelectedEap = "None"
   For $i = 1 To $asListEap[0]
      If GUICtrlGetState($aiEapEvent[$i]) = $GUI_ENABLE + $GUI_SHOW And GUICtrlRead($aiEapEvent[$i])
= $GUI_CHECKED Then
         $sSelectedEap = $asListEap[$i]
      EndIf
   Next
   $sSelectedRad = "None"
   For $i = 1 To $asListRADIUS[0]
      If GUICtrlGetState($aiRadEvent[$i]) = $GUI_ENABLE + $GUI_SHOW And GUICtrlRead($aiRadEvent[$i])
= $GUI_CHECKED Then
         $sSelectedRad = $asListRADIUS[$i]
      EndIf
   Next

   $sPDU_Enabled = _CheckPduComms($sPDU_IP) ; added 8.3.2009
   If $sPDU_Enabled = "YES" Then ;
      $iMsgBoxAnswer = MsgBox(33, "Final confirmation...", "You happy with your selections?")
      $iPDU_Port = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "PDU_Port", "NotFound") ; Get
correct port from ApIni-file
      $apip = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "IP", "NotFound")
; $aAPs_and_Ports[$i][0] = AP_Name, $aAPs_and_Ports[$i][1] = Port, $aAPs_and_Ports[$i][2] = Button
@GUI_CTRLID; $aAPs_and_Ports[$i][3] = Button State
      If $iPDU_Previous <> $iPDU_Port Then
         ;Else ; if same AP is used again, no need to shut it down
         If $iPDU_Previous <> 0 Then ; $iPDU_Previous = 0 in startup
                        _TogglePdu($iPDU_Previous, "Off")
```

```
            ;_TogglePduBtnState($iCtrlId);
         EndIf

         _TogglePdu($iPDU_Port, "On")

         $iPDU_Previous = $iPDU_Port
         _APC_GetState()
         ; tähän SetProgressOn, jotta AP:lla on aikaa bootata lisätty 29.10.2007
         ProgressOn("Booting up... " & $sSelectedAp, "Please wait...", $sSelectedAp & " is booting
up...")
         $i = 1

         Do
            $iIsApOnLine = Ping($apip, 5000) ; returns 0, when not online, when online returns
roundtrip time
            $i = $i + 1
            ProgressSet($i * 10, "Booting...")
            If $i = 8 Then
               $i = 1
            EndIf
         Until $iIsApOnLine Or $i = 7
         ProgressSet(90, "Hold on...", "Almost there, please be patient...")
         For $j = 1 To 10
            Sleep(1000) ; 10 sec delay may need to be adjusted due to slow boot of some APs
            ProgressSet(90 + $j & "Hold on...", 10 - $j & " secs to go...", "Almost there, please be
patient...")
         Next
         ProgressOff()
      EndIf
      If Not Ping($apip, 4000) Then
         MsgBox(33, "Access Point " & $sSelectedAp, "Access Point " & $sSelectedAp & " in Port: " &
$iPDU_Port & @CRLF & @CRLF & "is not responding. Please check cabling etc.")
      EndIf
   Else
      ; Prompt user to switch on AP if PDU is not used
      $iMsgBoxAnswer = MsgBox(33, "One more thing...", "Before continuing, make sure Access Point" &
@CRLF & @CRLF & $sSelectedAp & @CRLF & @CRLF & "is powered and functional")
   EndIf ; $sPDU_Enabled
   ; continue setting up AP
   Select
      Case $iMsgBoxAnswer = 1
         $iRestoreApCounterValue = _RestoreAP() ; _RestoreAP() returns zero, if NOK
         If $iRestoreApCounterValue > 0 Then; returns something, when ok, zero if failure
            _SetRadius()
            _UpdateListView()
            ;MsgBox(0, "$iRestoreApCounterValue", "$iRestoreApCounterValue = " & $iRestoreApCounter-
Value)
         EndIf
      Case $iMsgBoxAnswer = 2 ;Cancel, dang... Back to square 1 with previously selected options

   EndSelect ;MsgBox
EndFunc   ;==>_ButtonGoClick

Func MainClose()
   If $iIsListSaved = 0 Then
      Btn_Save_ListClick()
   EndIf
   _AllPortsOff()
   Exit
EndFunc   ;==>MainClose
Func MainMinimize()
```

```
#EndRegion - Main Window Button Clicked - functions

Func _ToggleApState($iEvent) ; Event number as input
   ; if input is in range 4 - 6 = 802button
   ; if input is in range 9 - 15 = AirButton

   $iSetAp = 0
   $iIndex = _ArraySearch($ai802Event, $iEvent);, 0, 0, 0, 1)
   If @error Then ; not in $ai802Event, but in $aiAirEvent, do capa check for aps
      $iIndex = _ArraySearch($aiAirEvent, $iEvent); $asListAir[0]
      $sPreviousAp = 0
      If $sSelected802 > 4 Then ; if > 4 then in 802.1n mode
      Else
         For $i = 1 To $asListAP[0] ;To 1 Step -1
            ;MsgBox(0, "$asApCapa[$i][$iIndex]", $asApCapa[$i][$iIndex],2)
            ;#cs
            If StringStripWS($asApCapa[$i][$iIndex], 8) = "" Then
; if selected air-mode is not supported by the ap -> disable
               GUICtrlSetState($aiApEvent[$i], $GUI_DISABLE)
; if disable AND selected, move selection to next available (or first available)
               If GUICtrlRead($aiApEvent[$i]) = $GUI_CHECKED Then
                  GUICtrlSetState($aiApEvent[$i], $GUI_UNCHECKED)
                  $sPreviousAp = $i
               EndIf
            Else ; air-mode supported by AP
               GUICtrlSetState($aiApEvent[$i], $GUI_ENABLE) ; + $GUI_CHECKED)
               ;MsgBox(0, "$asListAP", $asListAP[$i] & " = " & GUICtrlRead($aiApEvent[$i]),1)
            EndIf
         Next
         For $i = 1 To $asListAP[0]
            If GUICtrlRead($aiApEvent[$i]) = $GUI_CHECKED Then ; check if any checked
               $iSetAp = $i
            EndIf
         Next

         If $iSetAp = 0 Then
            For $i = $asListAP[0] To 1 Step -1
               If GUICtrlGetState($aiApEvent[$i]) = $GUI_ENABLE + $GUI_SHOW Then; 80 = 64 + 16
                  GUICtrlSetState($aiApEvent[$i], $GUI_CHECKED)
               EndIf
            Next
         EndIf

      EndIf
   Else ; Found in $ai802Event
      ;#cs
      If $iIndex > 1 Then
         ;MsgBox(0, "$ai802Event", "802.11n-mode, enable only 802.11n capable APs",2)
         For $i = 1 To $asListAP[0]
            ;MsgBox(0,"$asApCapa[$i][$asListAir[0]+1", $asApCapa[$i][$asListAir[0]+1])

            If StringStripWS($asApCapa[$i][$asListAir[0] + 1], 8) = "" Or String-
StripWS($asApCapa[$i][$asListAir[0] + 2], 8) = "" Then ; If 802.1x prefix not found, then disable
               GUICtrlSetState($aiApEvent[$i], $GUI_DISABLE + $GUI_UNCHECKED)
            Else
               GUICtrlSetState($aiApEvent[$i], $GUI_ENABLE)
            EndIf
         Next
         For $i = 1 To $asListAP[0]
            If GUICtrlRead($aiApEvent[$i]) = $GUI_CHECKED Then ; check if any checked
```

```
               $iSetAp = $i
            EndIf
         Next

         If $iSetAp = 0 Then
            For $i = $asListAP[0] To 1 Step -1
               If GUICtrlGetState($aiApEvent[$i]) = $GUI_ENABLE + $GUI_SHOW Then; 80 = 64 + 16
                  GUICtrlSetState($aiApEvent[$i], $GUI_CHECKED)
               EndIf
            Next
         EndIf
      Else
         ;MsgBox(0, "$ai802Event", "B/G-mode, enable all APs, unless some EAP prevents",2)
         For $i = 1 To $asListAP[0] ; if b/g pressed -> enable
            GUICtrlSetState($aiApEvent[$i], $GUI_ENABLE)
         Next
      EndIf
      ;#ce
   EndIf

EndFunc   ;==>_ToggleApState

Func _ToggleEapState($sEapState) ; if enable, need to check if air enterprise & which radius, use
enable also selectively (which radius -> may disable)

   For $i = 1 To $asListEap[0]
      GUICtrlSetState($aiEapEvent[$i], $sEapState)
   Next

EndFunc   ;==>_ToggleEapState

Func _ToggleRadiusState($sRadiusState) ; if enable, need to check eap-mode & if air = enterprise
   ;#cs
   If $sRadiusState = $GUI_ENABLE Then
      For $i = 1 To $asListEap[0] ; read which eap selected
         If GUICtrlRead($aiEapEvent[$i]) = $GUI_CHECKED Then
            $sSelectedEap = $i ; find selected eap
         EndIf
      Next
      $sSelectedRad = 0
      For $j = 1 To $asListRADIUS[0]
         If StringUpper($asRadCapa[$j][$sSelectedEap]) = "YES" Then
            GUICtrlSetState($aiRadEvent[$j], $sRadiusState)
            If $sSelectedRad = 0 Then
               GUICtrlSetState($aiRadEvent[$j], $GUI_CHECKED) ; check first matching
               $sSelectedRad = $j
            EndIf
         Else ; disable radius, if eap-type not supported
            GUICtrlSetState($aiRadEvent[$j], $GUI_DISABLE)
            If GUICtrlRead($aiRadEvent[$j]) = $GUI_CHECKED Then
               GUICtrlSetState($aiRadEvent[$j], $GUI_UNCHECKED)
               $sPreviousRad = $j
            EndIf
         EndIf
      Next

   Else
      For $j = 1 To $asListRADIUS[0]
         GUICtrlSetState($aiRadEvent[$j], $sRadiusState)
      Next
   EndIf
EndFunc   ;==>_ToggleRadiusState
```

```
Func _Debug_Events()

   Global $tEventArrays
   ; 802
   $tEventArrays = "$ai802Event" & @CRLF
   For $i = 0 To $iNumOf802Mode - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $ai802Event[$i] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$asList802Mode" & @CRLF
   For $i = 0 To $iNumOf802Mode - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $asList802Mode[$i] & @CRLF
   Next
   ; Air
   $tEventArrays = $tEventArrays & @CRLF & "$aiAirEvent" & @CRLF
   For $i = 0 To $iNumOfAir - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aiAirEvent[$i] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$asListAir" & @CRLF
   For $i = 0 To $iNumOfAir - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $asListAir[$i] & @CRLF
   Next
   ; AP
   $tEventArrays = $tEventArrays & @CRLF & "$aiApEvent" & @CRLF
   For $i = 0 To $iNumOfAP - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aiApEvent[$i] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$asListAP" & @CRLF
   For $i = 0 To $iNumOfAP - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $asListAP[$i] & @CRLF
   Next
   ; EAP
   $tEventArrays = $tEventArrays & @CRLF & "$aiEapEvent" & @CRLF
   For $i = 0 To $iNumOfEAP - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aiEapEvent[$i] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$asListEap" & @CRLF
   For $i = 0 To $iNumOfEAP - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $asListEap[$i] & @CRLF
   Next
   ; Radius
   $tEventArrays = $tEventArrays & @CRLF & "$aiRadEvent" & @CRLF
   For $i = 0 To $iNumOfRAD - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aiRadEvent[$i] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$asListRADIUS" & @CRLF
   For $i = 0 To $iNumOfRAD - 1
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $asListRADIUS[$i] & @CRLF
   Next
   ; PDU Ports
   $tEventArrays = $tEventArrays & @CRLF & "$aAPs_and_Ports[$i][2]" & @CRLF
   For $i = 0 To $iNumOfPduPorts
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aAPs_and_Ports[$i][2] & @CRLF
   Next
   $tEventArrays = $tEventArrays & @CRLF & "$aAPs_and_Ports[$i][3] Note Array Size" & @CRLF ;
$asApcPorts[][]
   For $i = 0 To $iNumOfPduPorts
      $tEventArrays = $tEventArrays & "[" & $i & "]" & "|" & $aAPs_and_Ports[$i][0] & " | " &
$aAPs_and_Ports[$i][3] & @CRLF
   Next

   _SaveTxt($tEventArrays, "EventArrays.txt")
```

```
   Exit
EndFunc   ;==>_Debug_Events

Func _SaveTxt($tTextToSave, $sSaveFile)
              $sListFileName = FileSaveDialog("Save ", @ScriptDir, "Text Documents (*.txt)", 16,
$sSaveFile)
; option 16 = dialog remains until valid path/file selected & prompts for overwrite, if file exists

   If @error Then
      MsgBox(48, "Save File", "Save cancelled!")
      Return 0 ; return $iIsListSaved
   Else
      $file = FileOpen($sListFileName, 2) ; Check if file opened for writing OK
      If $file = -1 Then
         MsgBox(48, "Error", "Unable to open file.")
         Exit
      EndIf
      FileWrite($file, $tTextToSave)
      FileClose($file)

      Return 1 ; return $iIsListSaved
   EndIf
EndFunc ; ==>  _SaveTxt()

Func _RestoreAP() ; returns 0 if failure?

   If StringInStr($sSelected802, "5") Then ; 5GHZ 802.11n
      $s802Prefix = StringStripWS(IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "802.11n_5", "Not
Found"), 8) ; Read 802.11n 5GHz file prefix from ini-file
   Else
      If StringInStr($sSelected802, "b/g") Then
         $s802Prefix = ""
      Else
         $s802Prefix = StringStripWS(IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "802.11n_2",
"Not Found"), 8) ; Read 802.11n 2.4GHz file prefix from ini-file
      EndIf
   EndIf
   $sRestoreFile = $s802Prefix & StringStripWS(IniRead($sLocalPath & $sApIniFile, $sSelectedAp,
$sSelectedAir, "Not Found"), 8) ; read file name from ini-file and add 802.11n prefix, if needed

   While ProcessExists("RestoreAp.exe") ;terminate old hanging processes
      ProcessClose("RestoreAp.exe")
   WEnd
   $iPidRestoreAp = Run(@ComSpec & " /c " & 'RestoreAp.exe ' & $sSelectedAp & ' ' & $sRestoreFile,
@ScriptDir, @SW_HIDE, $STDOUT_CHILD)

   Sleep(2000)
   $iRestoreApCounter = 0
   While ProcessExists($iPidRestoreAp)
      ;GUICtrlSetData($iCurrentLvItem,StdoutRead($iPidRestoreAp))
      Sleep(1000)
      $iRestoreApCounter = $iRestoreApCounter + 1
      If $iRestoreApCounter = 180 Then ; if restore process gets hang, then kill it
         ProcessClose($iPidRestoreAp)
         $iRestoreApCounter = 0
         ExitLoop
      EndIf
   WEnd
   Return ($iRestoreApCounter)
EndFunc   ;==>_RestoreAP

Func _CheckPduComms($sPDU_IP)
   If Not Ping($sPDU_IP, 3000) Then
```

```
      Return ("No")
   Else
      Return ("YES")
   EndIf
EndFunc   ;==>_CheckPduComms

Func _SetRadius()
   MsgBox(0, "SetRadius", "Set RADIUS to " & $sSelectedRad, 2)
   If ShellExecuteWait("SetRadius ", $sSelectedRad, $sLocalPath) Then
      $iTime = @HOUR & ":" & @MIN & " |" & @MDAY & "." & @MON & " " & @YEAR
      $sErrorItem = "Error in Radius-setup: " & $iTime & "|" & $sSelected802 & "|" & $sSelectedAir &
"|" & $sSelectedAp & "|" & $sSelectedEap & "|" & $sSelectedRad ; collect setup & time to oneliner
      $iErrorFile = FileOpen("Wizard-error.txt",1)
      FileWrite($iErrorFile, $sErrorItem)
      FileClose($iErrorFile)
      MsgBox(16, "Error in RADIUS Setup", "Radius error has occurred!")
   EndIf
EndFunc   ;==>_SetRadius

Func _UpdateListView()
   ;$ListView1
   $iUpdateListViewOK = 0
   $iTime = @HOUR & ":" & @MIN & " |" & @MDAY & "." & @MON & " " & @YEAR
   $sListViewItem = $iTime & "|" & $sSelected802 & "|" & $sSelectedAir & "|" & $sSelectedAp & "|" &
$sSelectedEap & "|" & $sSelectedRad ; collect setup & time to oneliner
   ;$iUpdateListViewOK = _GUICtrlListViewInsertItem($ListView1, 0, $sListViewItem) ; insert new item
to top of ListView
   ;MsgBox(0, "$sListViewItem", $sListViewItem)
   $iCurrentLvItem = GUICtrlCreateListViewItem($sListViewItem, $ListView1) ; insert new item to end
of ListView
   If $iFirstLvItem = 0 Then
      $iFirstLvItem = $iCurrentLvItem
   EndIf
   If $iCurrentLvItem <> 0 Then ; if not error
      $iUpdateListViewOK = 1
      $iIsListSaved = 0
   Else
      $iUpdateListViewOK = 0
   EndIf

EndFunc   ;==>_UpdateListVie
```

Liite 2: RestoreAP –ohjelman lähdekoodi

```
; RestoreAp($sAP,$sRestoreFile)
; Uses AP-INI -file, returns nothing
; AP ini-file can be found from VerificationWizard.ini
; [AP]
; APSetupInfo = APSetupInfo.ini
; Jukka Issakainen, Quality Assurance Services, 2009

#include <Constants.au3>
#include <array.au3>
#include<ie.au3>

Global $sAP, $sRestoreFile
Const $sIniFile = "VerificationWizard.ini"
Opt("SendkeyDelay",20)

$DEBUG_iAccessPointOff = 0
$DEBUG_iRadiusOff = 0

$sLocalPath = @ScriptDir & "\" ; Add trailing backslash
$sApIniFile = IniRead($sLocalPath & $sIniFile, "AP", "APSetupInfo", "NotFound") ; Read the name of
APinifile from Wizard.ini
$sAPRestorePath = IniRead($sLocalPath & $sApIniFile, "AP", "RestorePath", "NotFound"); Read from
ApINI-file, not anymore VerificationWizard.ini

FileInstall("3cserver.exe", $sLocalPath)
FileInstall("3cserver.ini", $sLocalPath)
FileInstall("plink.exe", $sLocalPath)

If $cmdline[0] > 0 Then ; Selected AP and restorefile as input, main prg takes care of matching
selected mode to restorefile

   $sSelectedAp = ""
   $iEndOfInput = $cmdline[0]
   For $i = 1 to $iEndOfInput -1
      $sSelectedAp = $sSelectedAp & $cmdline[$i] & " "
   Next
   $sRestoreFile = $cmdline[$iEndOfInput]
   ;MsgBox(0, "Parametrit", "$sSelectedAp = " & $sSelectedAp & @CRLF & "$sRestoreFile = " &
$sRestoreFile)
   $sAPRestoreFolder = $sAPRestorePath & IniRead($sLocalPath & $sApIniFile, $sSelectedAp,
"RestoreFolder", "NotFound")
   $aTelnetParam = StringSplit($sRestoreFile, ",")
; This function reads from $sRestoreFile separated parameters by comma, if Telnet > only one
parameter = restorefile
   AutoItSetOption("WinTitleMatchMode", 2)
; set title matching to sub-string, no need for whole title
   ;_ArrayDisplay($aTelnetParam, "$aTelnetParam")
   $sAP_IP = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "IP", "NotFound")
   $sGoRestore = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "GoRestore", "NotFound")
   $sUserID = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "UserID", "NotFound")
   $sPassword = IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "Password", "NotFound")

   If $sAP_IP = "Not Found" Or $sGoRestore = "Not Found" Or $sUserID = "Not Found" Or $sPassword =
"Not Found" Or $sAPRestoreFolder = "Not Found" Or $sRestoreFile = "Not Found" Then
      MsgBox(0, "Error", "Parameter not found, check " & $sApIniFile & " -file!" & @CRLF & @CRLF &
"Access Point " & $sSelectedAp & " might not support selected mode")
   EndIf

   If StringRight($sAPRestoreFolder, 1) <> "\" Then; Check that path contains a backslash
      $sAPRestoreFolder = $sAPRestoreFolder & "\"
   EndIf
   $sRestoreFile = $sAPRestoreFolder & $sRestoreFile ; combine path & restorefile
```

jatkuu

```
;  MsgBox(0, "AccessPoint Setup values", "$AP= " & $sSelectedAp & @CRLF & "$sAP_IP= " & $sAP_IP &
$sGoRestore & @CRLF & "$sUserID= " & $sUserID & @CRLF & "$sPassword =" & $sPassword & @CRLF &
"$sRestoreFile =" & $sRestoreFile)
If $DEBUG_iAccessPointOff = 0 Then ; $DEBUG_iAccessPointOff = 1 for debugging
      ; tähän tsekki onko ny varmasti oikea AP valittu eli vastaako pingiin
      $iIsApOnLine = Ping($sAP_IP, 4000) ;

      If $iIsApOnLine Then ; Is AP online?

         If $DEBUG_iRadiusOff = 0 Then ; if not in any debugmode, then set Progressbar
            ;ProgressOn("Access Point Configuration Restore", "Please wait...", "0 percent", -1, -1,
16) ; 16 = window can be moved
            ;AdlibEnable("_ProgressBar", 500) ; Call _ProgressBar every 500ms
         EndIf ; set ProgressBar
         If $aTelnetParam[0] = 1 And $aTelnetParam[1] <> "set" And Not StringInStr($sSelectedAp,
"Apple") Then ; wep interface used, if not telnet or SSH or Apple (-> Airport.exe)

            ;;ShellExecute("iexplore.exe", $sAP_IP & $sGoRestore)
            $oIE =_IECreate($sAP_IP & $sGoRestore)
            Sleep(3000)

            $sSelectedAp = StringStripWS($sSelectedAp, 8)
; to strip all WhiteSpaces away from AP name to avoid any misconfig

            ;#cs
Select
Case $sSelectedAp = "A-LinkWNAP";
      $oIE =_IEAttach("Connect","DialogBox")
      $oDoc = _IEDocGetObj($oIE)
      $oIE =_IEAttach("Save/Reload","Title")
      Sleep(3000); [CLASS:Internet Explorer_Server; INSTANCE:1]
      ControlClick("Save/Reload","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,220,164)
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      Sleep(3000)
      WinWaitActive($sAP_IP)
      Sleep(1000)
      Send("{TAB 8}" & "{ENTER}")
      WinWaitActive("Save/Reload Setting")
      WinClose("Save/Reload Setting")

Case $sSelectedAp = "Cisco1231"
      Send($sUserID & "{TAB}" & $sPassword & "{ENTER}")
      Sleep(2000)
      WinWaitActive("System Software - System Configuration")
      Sleep(7000)
      Send("+{TAB 10}" & $sAPRestoreFolder & $sRestoreFile & "+{TAB}{enter}")
; insert filename & go to "Load" -button
      WinWaitActive("Microsoft")
      Send("{enter}")
      Sleep(10000) ; wait for boot count-down timer
      WinWaitNotActive("System Restarting Now") ; wait for Cisco to boot-up
      Sleep(5000) ; Wait for other processes to startup in AP
      $iPing = Ping($sAP_IP)
      While @error = 1 ; host is offline
         Sleep(1000)
         $iPing = Ping($sAP_IP)
      WEnd
      WinWaitActive("Cisco")
      WinClose("Cisco") ; Close Browser after restore

               ;#ce
Case $sSelectedAp = "BelkinN1Vision";
      WinWaitActive("Login")
      Sleep(3000)
      Send("{TAB 2}" & "{ENTER}")
      WinWaitActive("Restore")
      Sleep(3000) ;
```

```
      ControlClick("Restore","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,370,184)
      ; Uses ControlClick Coordinates to navigate to input field
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Explorer","continue")
      Send("{ENTER}")
      WinWaitActive("Explorer", "90") ; 90s info window
      Send("{ENTER}")
      WinWaitActive("Setup Home")
      WinClose("Setup Home")

   Case $sSelectedAp = "BuffaloWZR-AG300NH";
      WinWaitActive("Connect")
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      WinWaitActive("AirStation Settings")
      Sleep(10000)
      ControlClick("AirStation Settings","","[CLASS:Internet Explorer_Server; IN-
      STANCE:1]","left",1,357,323)
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      Sleep(120000) ; estimated boot time 80s
      WinClose("AirStation Settings")

   Case $sSelectedAp =  "D-LinkDIR-655" ;
      WinWaitActive("Login")
      Sleep(2000)
      ControlClick("Login","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,580,237)
; Click Login-button
      Sleep(2000)
      WinWaitActive("Setup / Internet")
      Sleep(2000)
      Send("{TAB}" & $sAP_IP & $sGoRestore & "{ENTER}")
      WinWaitActive("Tools / System")
      Sleep(2000)
      ControlClick("Tools / System","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,350,395);
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      Sleep(10000)
      If WinGetTitle("Restore Success") Then
         ControlClick("Restore Success","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,156,222);
      Else
         WinWaitActive("Tools / System")
         Sleep(3000)
         ControlClick("Tools / System","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,410,552); Reboot the system
         Sleep(2000)
         WinWaitActive("Internet Explorer")
         Sleep(1000)
         Send("{ENTER}")
      EndIf

      WinWaitActive("Login")
      WinClose("Login")

   Case $sSelectedAp = "LinksysWRT54GS" ;
      WinWaitActive("Connect")
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      Sleep(2000)
      WinWait("Config Management")
      Sleep(2000)
      ControlClick("Config Management","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,373,300);
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      Sleep(3000)
```

```
      Winwait("restore.cgi")
      Sleep(5000)
      ControlClick("restore.cgi","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,391,237); Succesful OK-btn
      WinWait("Basic Setup")
      Sleep(3000)
      WinClose("Basic Setup")

Case $sSelectedAp = "LinksysWRT350N";
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Management") ; wait for this window become active
      Sleep(3000)
      Send("+{TAB 11}" & $sRestoreFile); error unmatched <PID>
      Sleep(1000)
      Send("+{TAB}" & "{ENTER}")
      WinWaitActive("Internet Explorer") ; Confirmation window
      Send("{ENTER}") ; close
      WinWaitActive("Basic Setup")
      WinClose("Basic Setup")

Case $sSelectedAp = "LinksysWRT610N" ;
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Restore Configurations") ; wait for this window become active
      Sleep(3000)
      Send("{TAB 8}" & $sRestoreFile & "{TAB 2}" & "{ENTER}")
; go to file field, input file and got to Restore
      WinWaitActive("restore.cgi")
      Sleep(3000)
      Send("{TAB 8}" & "{ENTER}")
      WinWaitActive("Internet Explorer") ; do you want to close the the tab
      Sleep(3000)
      Send("{ENTER}") ; close

   Case $sSelectedAp = "Netwjork54Mbps" ;
      ; Netwjork uses 3Com TFTP-server for restore, it MUST be located in Script-dir
      ProcessClose("3CServer.exe") ; just in case....
      WinWaitActive("Connect")
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      Sleep(3000)
      ShellExecute($sLocalPath & "3CServer.exe","","","",@SW_HIDE)
      WinWaitActive("3CServer")
      Sleep(3000)
      WinActivate("System |")
      Sleep(2000)
      FileCopy($sRestoreFile,$sAPRestoreFolder & "backup.bin",1) ; overwrite with no questions
      ; easier to do like this, as file input field is length-limited
      ControlClick("System |","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,325,170);
TFTP-server's IP
      Sleep(2000)
      Send("{BS 20}")
      Sleep(1000)
      Send(@IPAddress1)
      Sleep(1000)
      ControlClick("System |","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,108,225);
Click restore-button
      WinWaitActive("Internet Explorer")
      Send("{ENTER}")
      While Not StringInStr(ControlGetText("System |","","[CLASS:Edit; In-
stance:1]"),"system_reboot.asp") ; wait for Reboot-button
         Sleep(1000)
      WEnd
      Sleep(3000)
      ControlClick("System |","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,156,142);
press Reboot-button
      WinWaitActive("Internet Explorer")
      Send("{ENTER}")
      WinWaitActive("54M")
```

```
      WinClose("54M")
      ProcessClose("3CServer.exe") ; then no "Are you sure window"

   Case $sSelectedAp = "Thomson585" ;
      WinWaitActive("Restore")
      Sleep(3000)
      ControlClick("Restore","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,580,585);
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Internet Explorer")
      Send("{ENTER}")
      WinWaitActive("Home")
      Sleep(3000)
      WinClose("Home")

   Case $sSelectedAp = "TelewellTW-EA515" ;
      WinWaitActive("Connect")
      Send($sUserID & "{TAB}" & $sPassword & "{TAB 2}" & "{ENTER}")
      WinWaitActive("config/index.html")
      Sleep(2000)
      ControlClick("config/index.html","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,300,300);
      Sleep(1000)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      Sleep(3000)
      While Not WinGetTitle("3G/ADSL2+")
         Sleep(10000)
                        WEnd
      WinClose("3G/ADSL2+")

   Case $sSelectedAp = "ZyXELNBG-415N" ;
      WinWaitActive("Login")
      Sleep(2000)
      Send($sPassword & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Router Configuration")
      Sleep(2000);
      Send("{TAB}" & $sAP_IP & $sGoRestore & "{enter}")
      WinWaitActive("TOOLS / System")
      Sleep(2000) ;
      ControlClick("TOOLS / System","","[CLASS:Internet Explorer_Server; IN-
STANCE:1]","left",1,284,306)
      Send($sRestoreFile & "{TAB 2}" & "{ENTER}")
      WinWaitActive("Success")
      Send("{TAB 8}" & "{ENTER}")
      WinWaitActive("Internet Explorer")
      Send("{ENTER}")
      WinWaitActive("Login")
      WinClose("Login")

   Case $sSelectedAp = "ZyXelP-661HW-D1" ;
      WinWaitActive("Welcome")
      Sleep(2000)
      Send($sPassword & "{TAB}" & "{ENTER}")
      WinWaitActive("Welcome")
      Sleep(5000);
      Send("{TAB}" & $sAP_IP & $sGoRestore & "{ENTER}")
      Sleep(2000)
      ControlClick("Welcome","","[CLASS:Internet Explorer_Server; INSTANCE:1]","left",1,160,214)
      Sleep(5000)
      Send("{TAB 11}" & $sRestoreFile & "{TAB 2}" & "{ENTER}")
      While  Not StringInStr(ControlGetText(".::","", "[CLASS:Edit; INSTANCE:1]"), "home.html")
         Sleep(10000)
      WEnd
      WinClose(".::")
      EndSelect
         ;#ce
Else ; telnet or SSH ; on first round SSH-fingerprint needs to be cached; using Plink for automated
conns
```

```
   If StringInStr($sSelectedAp, "Apple") Then ;
      $sSelectedAp = "AppleAirportExtreme"
      ProcessClose("APUtil.exe") ; Close AirPort Utility, if found running
 ; GoRestore contains Path & filename, in this case "C:\Program Files\Airport\APUtil.exe"
      ShellExecute($sGoRestore)
      WinWaitActive("AirPort Utility")
      Sleep(2000)
      ControlClick("AirPort Utility","","[CLASS:Button;Instance:3]")
      Sleep(5000)
      ; Check here, if password has been remembered
      If WinGetTitle("Enter Password") Then
         ControlSend("Enter Password","","[CLASS:Edit;Instance:1]", $sPassword)
         Sleep(1000)
         ControlClick("Enter Password","","[CLASS:Button;Instance:1]","left",1) ;
      EndIf
      WinWaitActive("AirPort Utility")
      Sleep(2000)
      Send("!f") ; Send Control -I for Import
      Sleep(1000)
      Send("i")
      WinWaitActive("Open")
      Sleep(2000)
      ControlSend("Open","","[CLASS:Edit;Instance:1]", $sRestoreFile)
; insert filename to inputfield
      Sleep(1000)
      ControlClick("Open","","[CLASS:Button;Instance:2]","left",1) ; Click Open button
      WinWaitActive("Import")
      Sleep(1000)
      ControlClick("Import","","[CLASS:Button;Instance:1]","left",1) ; Click OK
      WinWaitActive("AirPort Utility")
      Sleep(1000)
      ControlClick("AirPort Utility","","[CLASS:Button;Instance:7]","left",1) ; Click Update button
      WinWaitClose("Writing to the Apple") ; wait until restore completed
      Sleep(3000)
      WinWaitClose("Reading") ; wait until utility reads from AP the status
      WinWaitActive("AirPort Utility")
      Sleep(3000)
      $sStatus = ControlGetText("AirPort Utility","","[CLASS:Static;Instance:6]")
      If $sStatus = "Normal" Then
         ProcessClose("APUtil.exe")
      Else
         MsgBox(0, "ERROR has occurred", "Something wrong, AP status = " & $sStatus)
      EndIf
             Else
      Send("#r") ; Windows Run-key
      WinWaitActive("Run")

      If IniRead($sLocalPath & $sApIniFile, $sSelectedAp, "SSH", "NotFound") = "Yes" Then
; SSH or Telnet?
         Send($sLocalPath & "plink -ssh " & $sAP_IP & "{enter}") ;
      Else ; Telnet
         Send($sLocalPath & "plink -telnet " & $sAP_IP & "{enter}") ;
      EndIf ; SSH or Telnet
      WinActivate("Plink", "")
      Sleep(1000)
      Send($sUserID & "{enter}")
      Sleep(1000)
      Send($sPassword & "{enter}")
      Sleep(1000)
      Send("{enter}") ; jostain syystä tuli yksi '+'-merkki cmdriville, tää tuntui auttavan
      Sleep(1000)
      For $i = 1 To $aTelnetParam[0]
; loop until all parameters from INI-file are sent, $aTelnetParam[0] contains # of parameters
         ;MsgBox(0, "$aTelnetParam", $aTelnetParam[$i])
         Send(StringStripWS($aTelnetParam[$i], 3), 1) ; jos pisti samalle riville {enterin}, niin
'+'-merkki lähti hitoille esim (WPA+WPA2)-PSK
         Send("{enter}")
```

```
         Sleep(1000)
      Next
      Sleep(1000)
      EndIf; Apple of Telnet
   EndIf ; web interface used, if not telnet or SSH or Apple
Else ; Is AP online?
   If @error = 0 Then $problem = "Network Errors Occured "
   If @error = 1 Then $problem = "Host is Off-Line "
   If @error = 2 Then $problem = "Host is Unreachable "
   If @error = 3 Then $problem = "Incorrect Destination "
   If @error = 4 Then $problem = "Network Errors Occured "
   If $iIsApOnLine = "NotFound" Then $problem = "Error in INI-file"
   MsgBox(48, "Problem occurred !!!", "Selected Access Point does not respond." & @CRLF & @CRLF &
"Error is: " & @error & " = " & $problem & @CRLF & @CRLF & "Check your AP selection...")
   $iApPIDReturn = 0 ; error occurred
EndIf ; Is AP online?
   Else; $DEBUG_iAccessPointOff = 1 for debugging
      MsgBox(0, "AccessPoint DEBUG: Setup values", "$AP= " & $sSelectedAp & @CRLF & "$sAP_IP= " &
$sAP_IP & @CRLF & "$sGoRestore = " &  $sGoRestore & @CRLF & "$sUserID= " & $sUserID & @CRLF &
"$sPassword =" & $sPassword & @CRLF & "$sAPRestoreFolder =" & $sAPRestoreFolder & @CRLF &
"$sRestoreFile =" & $sRestoreFile)

EndIf ; $DEBUG_iAccessPointOff = 1
Else
      MsgBox(16, "Error occurred", "No input from Wizard, exiting",2)
      Exit
EndIf
```

Liite 3: SetRadius –ohjelman lähdekoodi

```
; SetRadius($sSelectedRad)
; returns what psexec returns
; 0 = ok, 1 = error
; Jukka Issakainen, 2009

#include <Constants.au3>
Const $sIniFile = "VerificationWizard.ini"
Global $sSelectedRad

If $cmdline[0] > 0 Then
   $sSelectedRad = ""
   $iEndOfInput = $cmdline[0]
   For $i = 1 to $iEndOfInput
      $sSelectedRad = $sSelectedRad & $cmdline[$i] & " "
   Next
   $sSelectedRad = StringStripWS($sSelectedRad,2) ; strip trailing white space
   $sLocalPath = @ScriptDir & "\" ; Add trailing backslash
   $sRadiusIniFile = IniRead($sLocalPath & $sIniFile, "RADIUS", "RadiusSetupInfo", "NotFound"); Read
the name of Radiusinifile from Wizard.ini
   $sLogonDomain = IniRead($sLocalPath & $sIniFile, "RADIUS", "TestUser_Logon_Domain", "NotFound")
   $sRadiusUser = IniRead($sLocalPath & $sRadiusIniFile, $sSelectedRad, "Username", "NotFound")
   $sRadiusUserPwd = IniRead($sLocalPath & $sRadiusIniFile, $sSelectedRad, "UserPwd", "NotFound")
   $sRadiusRestoreFile = IniRead($sLocalPath & $sRadiusIniFile, $sSelectedRad, "RestoreFile",
"NotFound")
   $sRadiusRestorePath = IniRead($sLocalPath & $sRadiusIniFile, "RADIUS", "RestorePath", "NotFound")
   $sRadiusProxy = "\\" & IniRead($sLocalPath & $sRadiusIniFile, "RADIUS", "RadiusProxy",
"NotFound")
   $sRadiusRestoreFile = IniRead($sLocalPath & $sRadiusIniFile, $sSelectedRad, "RestoreFile",
"NotFound")
   $iRadiusSetStatus = ShellExecuteWait("psexec.exe", $sRadiusProxy & " -u " & $sLogonDomain & "\" &
$sRadiusUser & " -p " & $sRadiusUserPwd & " netsh exec " & $sRadiusRestorePath & "\" &
$sRadiusRestoreFile, $sLocalPath, '', @SW_SHOW)
      If $iRadiusSetStatus <> 0 Then
      MsgBox(16, "ERROR", "Error was " & $iRadiusSetStatus,2 ) ; 0 = ok, 1 = error
      Exit 16
   EndIf

Else
   MsgBox(16, "Error occurred", "No input from Wizard, exiting",2)
   Exit 16
EndIf
```

Liite 4: PSExec-ohjelman lähdekoodi

```
#include <Constants.au3>
;#include <array.au3>
Const $sIniFile = "VerificationWizard.ini"
;[RADIUS]
$sLocalPath = @ScriptDir & "\" ; Add trailing backslash
$sRadiusIniFile = IniRead($sLocalPath & $sIniFile, "RADIUS", "RadiusSetupInfo", "NotFound"); Read
the name of Radiusinifile from Wizard.ini
$sRadiusRestorePath = IniRead($sLocalPath & $sRadiusIniFile, "RADIUS", "RestorePath", "NotFound")
$sRadiusProxy = "\\" & IniRead($sLocalPath & $sRadiusIniFile, "RADIUS", "RadiusProxy", "NotFound")
$sSelectedRad = "Microsoft IAS"
$sRadiusRestoreFile = IniRead($sLocalPath & $sRadiusIniFile, $sSelectedRad, "RestoreFile",
"NotFound")
;MsgBox(0, "mitä tuli valittua", "$sRadiusIniFile = " & $sRadiusIniFile & @CRLF & "$sRadiusRestore-
Path = " & $sRadiusRestorePath & @CRLF & "$sRadiusProxy = " & $sRadiusProxy & @CRLF &
"$sRadiusRestoreFile = " & $sRadiusRestoreFile)
$valst_2 = ShellExecuteWait("psexec.exe", $sRadiusProxy & " -u wlan-auth20\administrator -p iopwlan
netsh exec " & $sRadiusRestorePath & "\" & $sRadiusRestoreFile, $sLocalPath, '', @SW_SHOW)
            If $valst_2 <> 0 Then
      MsgBox(0, "ERROR", "Error was " & $valst_2 ) ; 0 = ok, 1 = error
      Exit
   EndIf
```

Liite 5: VerificationWizard.ini –tiedosto

```
; WLAN VerificationWizard.ini
; Contains settings and variables
;
; Author: Jukka Issakainen, 2005 - 2009
;


[AP]
APSetupInfo = APSetupInfo.ini

[EAP]
;Types = TLS/PEAPv0, PEAPv1, TTLS-EAP-MSChapv2, TTLS-MSChapV2, LEAP, SIM, AKA
;Types = TLS/PEAPv0, PEAPv1, TTLS-EAP-MSChapv2, TTLS-MSChapV2, LEAP, SIM
Types = TLS/PEAPv0, PEAPv1, TTLS, SIM, AKA

[Misc]
; these are just informational texts to show on screen, after AP-setup is completed
SSID = QAS_Verification
WEP64_Key = 1234567890, Key 2 (10 HEX digits)
WEP128_Key = 1234567890abcdef1234567890, Key 2 (26 HEX digits)
WPA(2)-PSK_Key = 12345678
TestUser = testi
TestUser_PWD = iopwlan
TestUser_Logon_Domain = WLAN-AUTH20

[RADIUS]
RadiusSetupInfo = RadiusSetupInfo.ini
```

Liite 6: APSetupinfo.ini –tiedosto

```
; Access Point Info file
; Contains settings and variables
;
; Author: Jukka Issakainen, 2005 - 2009
;
; PDU_Port = Physical AC outlet # in APC Switched Rack PDU
; Model =    Info abt AP model, not used, but clarifies ini-file
; Firmware = Info abt Firmware, not yet used
; IP =    IP-address of AP
; SSH = yes/no  If AP is capable of SSH
; 802.11n_2 = If AP supports 802.1n 2.4GHz mode, prefix for setupfile (e.g. WNAP_N2.4_)
; 802.11n_5 = If AP supports 802.1n 5GHz mode, prefix for setupfile (e.g. WNAP_N5_
; UserID =   User name of AP administrative account
; Password = Password of AP administrative account
; GoRestore =   Path to config restore page inside Access Point, if exists
; RestoreFolder = Folder used to store config-files, empty folder, if Telnet/SSH is used
; Open =     Open, no wep-key etc
; 802.1x=    802.1x with dynamic wep-key 128-bit and RADIUS
; WEP128 =   wep-key 128-bit, no RADIUS
; WPA-PSK =  WiFi Protected Access, Pre Shared Key using TKIP + AES (a.k.a mixed WPA
SOHO-mode), no RADIUS
; WPA =      WiFi Protected Access, RADIUS for EAP-types using TKIP + AES (a.k.a mixed
WPA Enterprise-mode)
; WPA2-PSK = WiFi Protected Access 2, Pre Shared Key using AES, no RADIUS
; WPA2 =     WiFi Protected Access 2, RADIUS for EAP-types using AES
; LEAP =     Cisco specific mode

[AP]
Models = A-Link WNAP, Apple Airport Extreme, Cisco 1231, Belkin N1 Vision, Buffalo WZR-
AG300NH, D-Link DIR-655, Linksys WRT54GS, Linksys WRT350N, Linksys WRT610N, Netwjork
54Mbps, Thomson 585, Telewell TW-EA515, ZyXEL NBG-415N, ZyXel P-661HW-D1
RestoreModes = Open, WEP128, WPA-PSK, WPA2-PSK, WPA, WPA2, 802.1x
RestorePath = D:\Jukan\automation\AP_Setups
802Modes = 802.11 b/g, 802.11n (2.4GHz), 802.11n (5GHz)

[PDU]
; PDU_Enabled = Yes or No
; New interface after FW upgrade 2.70 or newer: <password><space>-c
PDU_Enabled = yes
PDU_IP = 10.10.32.17
PDU_User = wizard
PDU_Pwd = wizard -c

;############################## AP SETUP starts ##################
[A-Link WNAP]
PDU_Port = 16
Model = WL524
Firmware = e2.04
IP = 10.10.32.151
SSH = no
802.11n_2 = WNAP_N2_
802.11n_5 = WNAP_N5_
UserID = admin
Password = admin
GoRestore = /saveconf.asp
RestoreFolder = \A-LinkWNAP
Open = A-Link_Open.dat
```

jatkuu

```
802.1x =
WEP64 =
WEP128 = A-Link_WEP128.dat
WPA-PSK = A-Link_wpa-psk_mixed.dat
WPA =  A-Link_WPA_enterprise_mixed.dat
WPA2-PSK =
WPA2 = A-Link_WPA2_enterprise.dat
LEAP =

[Apple Airport Extreme]
PDU_Port = 15
Model = 123
Firmware = 123
IP = 10.10.32.152
SSH = no
802.11n_2 = Apple_N2_
802.11n_5 = Apple_N5_
UserID =
Password = iopwlan
GoRestore = "C:\Program Files\Airport\APUtil.exe"
RestoreFolder = \Apple
Open = Apple_Open.baseconfig
802.1x =
WEP64 =
WEP128 =
WPA-PSK = Apple_wpa-psk_mixed.baseconfig
WPA =  Apple_wpa_mixed.baseconfig
WPA2-PSK = Apple_wpa2-psk.baseconfig
WPA2 = Apple_wpa2.baseconfig
LEAP =

[Belkin N1 Vision]
PDU_Port = 14
Model = DIR-655
Firmware = F5D8232-4_WW_1.00.15
IP = 10.10.32.153
SSH = no
802.11n_2  = Belkin_N2_
802.11n_5 = Belkin_N5_
UserID = admin
Password =
GoRestore = /setup.cgi?next_file=ut_prev.html
RestoreFolder = \BelkinN1Vision
Open = BelkinN1Vision_Open.conf
802.1x=
WEP64 =
WEP128 = BelkinN1Vision_WEP128.conf
WPA-PSK = BelkinN1Vision_wpa-psk_mixed.conf
WPA =
WPA2-PSK = BelkinN1Vision_wpa2-psk.conf
WPA2 =
LEAP =

[Buffalo WZR-AG300NH]
PDU_Port = 13
Model = WZR-AG300NH
Firmware = 1.49
IP = 10.10.32.154
SSH = no
802.11n_2  = Buffalo_N2_
```

```
802.11n_5 = Buffalo_N5_
UserID = root
Password =
GoRestore = /cgi-bin/cgi?req=tfr&id=47
RestoreFolder = \BuffaloWZR
Open = BuffaloWZR_Open.bin
802.1x=
WEP64 =
WEP128 = BuffaloWZR_WEP128.bin
WPA-PSK = BuffaloWZR_wpa-psk_mixed.bin
WPA =
WPA2-PSK = BuffaloWZR_wpa2-psk.bin
WPA2 =
LEAP =

[Cisco 1231]
PDU_Port = 12
Model = Air 1231G-K9
Firmware = 12.3(8)JEA
IP = 10.10.32.150
SSH = no
802.11n_2 =
802.11n_5 =
UserID = Cisco
Password = Cisco
GoRestore = /ap_system-sw_sysconfig.shtml
RestoreFolder = \Cisco1231
Open = Cisco1231_Open.txt
802.1x= Cisco1231_8021x.txt
WEP64 = Cisco1231_WEP64.txt
WEP128 = Cisco1231_WEP128.txt
WPA-PSK = Cisco1231_WPA-PSK_mixed.txt
WPA =  Cisco1231_WPA_mixed.txt
WPA2-PSK = Cisco1231_WPA2-PSK.txt
WPA2 = Cisco1231_WPA2.txt
LEAP = Cisco1231_WPA2_Funk.txt


[D-Link DIR-655]
PDU_Port = 11
Model = DIR-655
Firmware = 1.21EU
IP = 10.10.32.162
SSH = no
802.11n_2 = DIR655_N2_
802.11n_5 = DIR655_N5_
UserID = admin
Password =
GoRestore = /Tools/System.shtml
RestoreFolder = \DIR-655
Open = DIR-655_Open.gws.htm
802.1x=
WEP64 = DIR-655_WEP64.gws.htm
WEP128 = DIR-655_WEP128.gws.htm
WPA-PSK = DIR-655_WPA-PSK_mixed.gws.htm
WPA =  DIR-655_WPA_mixed.gws.htm
WPA2-PSK = DIR-655_WPA2-PSK.gws.htm
WPA2 = DIR-655_WPA2.gws.htm
LEAP =
```

```
[Linksys WRT54GS]
PDU_Port = 9
Model = WRT54GS
Firmware = 4.71.1
IP = 10.10.32.155
SSH = no
802.11n_2 =
802.11n_5 =
UserID =
Password = admin
GoRestore = /Backup_Restore.asp
RestoreFolder = \wrt54gs
Open = WRT54GS_Open.cfg
802.1x=
WEP64 =
WEP128 = WRT54GS_WEP128.cfg
WPA-PSK = WRT54GS_WPA2-psk_mixed.cfg
WPA = WRT54GS_WPA2_mixed.cfg
WPA2-PSK = WRT54GS_WPA2-psk.cfg
WPA2 = WRT54GS_WPA2.cfg
LEAP =

[Linksys WRT350N]
PDU_Port = 8
Model = WRT350N
Firmware = 2.00.19
IP = 10.10.32.163
SSH = no
802.11n_2 = WRT350_N2_
802.11n_5 = WRT350_N5_
UserID = admin
Password = admin
GoRestore = /setup.cgi?next_file=Administration.htm
RestoreFolder = \WRT350Nv2
Open = WRT350v2_open.cfg
802.1x=
WEP64 =
WEP128 = WRT350v2_WEP128_key1.cfg
WPA-PSK = WRT350v2_wpa-psk_mixed.cfg
WPA = WRT350v2_wpa.cfg
WPA2-PSK = WRT350v2_wpa2-psk.cfg
WPA2 = WRT350v2_wpa2.cfg
LEAP =


[Linksys WRT610N]
PDU_Port = 7
Model = WRT610N
Firmware = 1.00.02.10
IP = 10.10.32.156
SSH = no
802.11n_2 = WRT610_N2_
802.11n_5 = WRT610_N5_
UserID = admin
Password = admin
GoRestore = /Restore.asp
RestoreFolder = \WRT610N
Open = WRT610NV1_open.cfg
802.1x=
WEP64 =
WEP128 = WRT610NV1_WEP128.cfg
```

```
WPA-PSK = WRT610NV1_WPA-psk_mixed.cfg
WPA = WRT610NV1_WPA_mixed.cfg
WPA2-PSK = WRT610NV1_WPA2-psk.cfg
WPA2 = WRT610NV1_WPA2.cfg
LEAP =


[Netwjork 54Mbps]
; Netwjork uses 3ComTFTP-server for setup backup/restore,
; which MUST be located in Script-dir

PDU_Port = 6
Model = 54Mbps
Firmware = default
IP = 10.10.32.157
SSH = no
802.11n_2 =
802.11n_5 =
UserID = admin
Password = admin
GoRestore = /system_backup.asp
RestoreFolder = \Netwjork
Open = Netwjork_open.bin
802.1x=
WEP64 =
WEP128 = Netwjork_WEP128.bin
WPA-PSK = Netwjork_wpapsk.bin
WPA =
WPA2-PSK = Netwjork_wpa2psk.bin
WPA2 =
LEAP =

[Telewell TW-EA515]
PDU_Port = 5
Model = TW-EA515
Firmware = 5.53.S3.ds36
IP = 10.10.32.160
SSH = no
802.11n_2 =
802.11n_5 =
UserID = admin
Password = admin
GoRestore = /config
RestoreFolder = \TelewellEA515
Open = TelewellEA515_open.icf
802.1x=
WEP64 =
WEP128 = TelewellEA515_WEP128.icf
WPA-PSK = TelewellEA515_wpa-psk.icf
WPA =
WPA2-PSK = TelewellEA515_wpa2-psk.icf
WPA2 =
LEAP =

[Thomson 585]
PDU_Port = 4
Model = ST780
Firmware = 6.2.16.3
IP = 10.10.32.158
SSH = no
802.11n_2 =
```

```
802.11n_5 =
UserID = Administrator
Password =
GoRestore = /cgi/b/bandr/?be=0&l0=0&l1=1&tid=BACKUP_RESTORE
RestoreFolder = \Thomson585
Open = Thomson_585_Open.ini
802.1x=
WEP64 =
WEP128 = Thomson_585_WEP128.ini
WPA-PSK = Thomson_585_wpa-psk_mixed.ini
WPA =
WPA2-PSK = Thomson_585_wpa2-psk.ini
WPA2 =
LEAP =


[ZyXEL NBG-415N]
PDU_Port = 2
Model = NBG-415N
Firmware = 3.60(ZP.2)C0
IP = 10.10.32.164
SSH = no
802.11n_2 = Zyxel_N2_
802.11n_5 = Zyxel_N5_
UserID =
Password = 1234
GoRestore = /Tools_System.html
RestoreFolder = \ZyXel415N
Open = Zyxel415N_Open.gws
802.1x=
WEP64 =
WEP128 = Zyxel415N_WEP128.gws
WPA-PSK = Zyxel415N_wpa-psk_mixed.gws
WPA =  Zyxel415N_wpa_mixed.gws
WPA2-PSK = Zyxel415N_wpa2-psk.gws
WPA2 = Zyxel415N_wpa2.gws
LEAP =

[ZyXel P-661HW-D1]
PDU_Port = 1
Model = P-661HW-D1
Firmware = 3.40(AHQ.0)
IP = 10.10.32.161
SSH = no
802.11n_2 =
802.11n_5 =
UserID =
Password = 12345
GoRestore = /RestoreCfg.html
RestoreFolder = \ZyXel661
Open = P-661HW-D1_Open
802.1x=
WEP64 =
WEP128 = P-661HW-D1_WEP128
WPA-PSK = P-661HW-D1_WPA-PSK_mixed
WPA =  P-661HW-D1_WPA_mixed
WPA2-PSK = P-661HW-D1_WPA2-PSK
WPA2 = P-661HW-D1_WPA2
LEAP =
;########################## AP SETUP ends ################################
```

Liite 7: RadiusSetup.ini –tiedosto

```
; RADIUS Setup Info -file
; Contains settings and variables
;
; Author: Jukka Issakainen, 2005 - 2009
;
; NOTE! RADIUS' names in brackets '[]' MUST be same as in 'Servers' -line!!!
;
;
; Parameters used: (Check EAP-types, MUST be same as in VerificationWizard.ini -file)
; IP = server's IP-address
; Port = used RADIUS-port
; RestoreFile = Nestsh dumpfile name
; TLS/PEAPv0 = yes/no
; PEAPv1 = yes/no
; TTLS = yes/no
; PureTTLS = yes/no <- not currently used
; LEAP = yes/no <- not currently used
; SIM = yes/no
; AKA = yes/no
; KOE = yes/no <- for testing purposes


[RADIUS]
RestorePath = D:\wlan\ap_setups\radius
RadiusProxy = 10.10.32.10
Username = wizard
UserPwd = wizard
; Proxy address is needed for psexec, which will run e.g. netsh exec ias.set on remote

Servers = Microsoft IAS, Juniper Odyssey, Cisco ACS, FreeRADIUS, Nokia Test Network

;###################### RADIUS SETUP starts ########################################
;
; RADIUS *.set files are created with NETSH-command. It dumps current IAS-setting to a
file, which can be replayed
; to instantly change authentication settings
;
; NOTE!!!! Applicable to Windows 2003 Server ONLY!!!!

; To record current settings enter to command prompt: netsh aaaa dump > filename.set

; To Replay saved settings enter to command prompt: netsh exec filename.set
```

jatkuu

```
[Microsoft IAS]
IP = 10.10.32.20
Port = 1812
RestoreFile = ias.set
TLS/PEAPv0 = yes
PEAPv1 = no
TTLS = no
PureTTLS = no
LEAP = no
SIM = no
AKA = no
KOE = no

[Juniper Odyssey]
IP = 10.10.32.50
Port = 1814
RestoreFile = juniper.set
TLS/PEAPv0 = yes
PEAPv1 = yes
TTLS = yes
PureTTLS = yes
LEAP = yes
SIM = no
AKA = no
KOE = no

[FreeRADIUS]
IP = 10.10.32.13
Port = 1814
RestoreFile = freeradius.set
TLS/PEAPv0 = no
PEAPv1 = yes
TTLS = yes
PureTTLS = yes
LEAP = no
SIM = yes
AKA = no
KOE = no

[Cisco ACS]
IP = 10.10.32.50
Port = 1812
RestoreFile = acs.set
TLS/PEAPv0 = yes
PEAPv1 = yes
TTLS = no
PureTTLS = no
```

```
LEAP = yes
SIM = no
AKA = no
KOE = no

[Nokia Test Network]
RestoreFile = NTN.set
TLS/PEAPv0 = no
PEAPv1 = no
TTLS = no
PureTTLS = no
LEAP = no
SIM = yes
AKA = yes
KOE = no

;#################### RADIUS SETUP ends ###################################
```